Explorations in Design Space: Unconventional Electronics Design Through Artificial Evolution

Adrian Thompson, Paul Layzell, and Ricardo Salem Zebulum

Abstract—**Three hypotheses are formulated. First, in the "design space" of possible electronic circuits, conventional design methods work within constrained regions, never considering most of the whole. Second, evolutionary algorithms can explore some of the regions beyond the scope of conventional methods, raising the possibility that *better* designs can be found. Third, evolutionary algorithms can *in practice* produce designs that are beyond the scope of conventional methods, and that are in some sense better.**

A reconfigurable hardware controller for a robot is evolved, using a conventional architecture with and without orthodox design constraints. In the unconstrained case, evolution exploited the enhanced capabilities of the hardware. A tone discriminator circuit is evolved on an FPGA without constraints, resulting in a structure and dynamics that are foreign to conventional design and analysis. The first two hypotheses are true.

Evolution can explore the forms and processes that are natural to the electronic medium, and nonbehavioral requirements can be integrated into this design process, such as fault tolerance. A strategy to evolve circuit robustness tailored to the task, the circuit, and the medium, is presented. Hardware and software tools enabling research progress are discussed. The third hypothesis is a good working one: practically useful but radically unconventional evolved circuits are in sight.

Index Terms—**Design automation, electronics design, evolutionary algorithms, evolutionary theory, fault tolerance.**

I. Introduction

IMAGINE a *design space* [1] where each point in that space represents the design of an electronic circuit. All possible electronic circuits are there, given the component types available to an electronics engineer, and the technological restrictions on how many components there can be and how they can interact. In this metaphor, we loosely visualize the circuits to be arranged in the design space so that "similar" circuits are close to each other.

The design space is vast. There are oscillators and filters, finite-state machines, analog computers, parallel distributed systems, von Neumann computers, and so on. These nestle amongst the majority of circuits for which a use will never be found. In this paper, we investigate the following hypotheses.

H1) Conventional design methods can only work within constrained regions of design space. Most of the whole design space is never considered.

H2) Evolutionary algorithms can explore some of the regions in design space that are beyond the scope of conventional methods. In principle, this raises the possibility that designs can be found that are in some sense better.

H3) Evolutionary algorithms *in practice* can produce designs that are beyond the scope of conventional methods and are, in some sense, better.

Note that the truth of each hypothesis relies in part on the truth of the preceding ones. In Section II we attempt to verify the first hypothesis by characterizing what electronics designers normally do. Being essentially a work of anthropology, this is inevitably subjective and open to dispute. We judge that the basic conclusions are robust to variations in the exact details of how the designer's activities are conceptualized.

In the case studies of Sections III and IV, the second hypothesis is verified empirically, to a high degree of confidence. The hierarchical dependency of the hypotheses means that H1 is also strengthened.

In the final section, tools are presented for ongoing research to produce circuits through artificial evolution that are beyond the scope of conventional design, are in some sense better, and that are practically useful. This is the third hypothesis. From the experimental results, it is clear that there can be some special "niches" [1] for unusual circuits, but it remains to be seen how broadly they can be applied. We interpret our results as encouraging.

A. Algorithms

Most of the discussion applies equally to any evolutionary algorithm (EA), in fact to any search algorithm based on a process of "generate-and-test." Since our aim is to explore design space as freely as possible, we avoid incorporating heuristics into the search that make sweeping assumptions about the characteristics of desirable circuits. For performance superior to random search, however, some domain-specific search biases are necessary [2]. Hence evolutionary search is used, the bias of which (that future candidate circuits should be based on variations of the more successful earlier ones) makes minimal—but not negligible—assumptions as to the nature of the circuit itself.

Biases also arise, often unintentionally, from the circuit representation to which the operators are applied. Skews in the number of points representing each circuit (or type of circuit), and clustering of circuit types in the representation with respect to the operators, can both bias the searching of even a simple

Manuscript received February 15, 1999.

The authors are with the Centre for Computational Neuroscience and Robotics, and Centre for The Study of Evolution, School of Cognitive and Computing Sciences, University of Sussex, Brighton BN1 9QH, UK (e-mail: adrianth@cogs.susx.ac.uk; paulla@cogs.susx.ac.uk; ricardoz@cogs.susx.ac.uk).

Publisher Item Identifier S 1089-778X(99)07199-4.

$(1+1)$ evolution strategy [3]. For many EA's, the local gradient of fitness with respect to the operators can bias the direction of search, for example toward relatively "smooth" parts of the fitness landscape [4], [5]. Whether helpful or adverse for search efficacy, such biases are not easily avoided, but can sometimes be turned to specific uses such as giving graceful degradation properties to evolved circuits [6].

Later, when more of the uncharted territory of design space has been investigated, the performance of the EA might be improved by incorporating this heuristic knowledge into the circuit representation, or the variation and selection operators. In this paper we use basic genetic algorithms (GA's), often with very direct "genetic" encodings. This is to aid clarity in describing the experiments and does not imply that these GA's are particularly effective EA's for these tasks. Note that at any given time, an EA is searching over accessible variations rather than the entire design space [7]; no attempt is made at global search, but instead at the exploration of new regions.

B. The Objectives of Circuit Design

The objectives of circuit design can be divided into behavioral and nonbehavioral requirements.

Behavioral requirements define the desired interaction between the circuit and its external environment. One possibility is to describe the required behavior directly at the interface between the circuit and environment in the form of a relationship between inputs and outputs over time: direct behavioral requirements. Another is to define the desired behavior of a larger system in which the circuit is embedded: embedded behavioral requirements. An example of the latter would be to define the required behavior of a robot that the circuit controls, rather than defining the input/output relationship of the controller itself. The need to keep electromagnetic emissions within acceptable limits is often thought of at both direct and embedded levels: guidelines for the emissions of subsystems, combined with principles for systems-level design to guarantee overall performance [8].

Nonbehavioral requirements define the resources that are available to the circuit and the environmental conditions under which it must continue to operate for some minimum lifetime at a given maximum failure rate. Examples of resources are size (silicon or circuit-board area), weight, power consumption, and construction cost. Examples of possibly relevant environmental conditions are temperature, output load, power-supply voltage, fabrication variations, defects (needing fault tolerance), electromagnetic interference (EMI), humidity, and mechanical vibration. The combinations and ranges of such environmental variables under which the circuit must meet its behavioral requirements can be thought of as defining its *operational envelope*: a set of points in a space spanned by the environmental variables. As an operational envelope consists partly of ranges of environmental variables, it can be visualized usefully in terms of regions for correct operation (Fig. 1). In general, an operational envelope may be of arbitrary structure, so this visual metaphor should be used with care.

We describe a circuit that performs adequately throughout the operational envelope defined for its task as being *robust*.

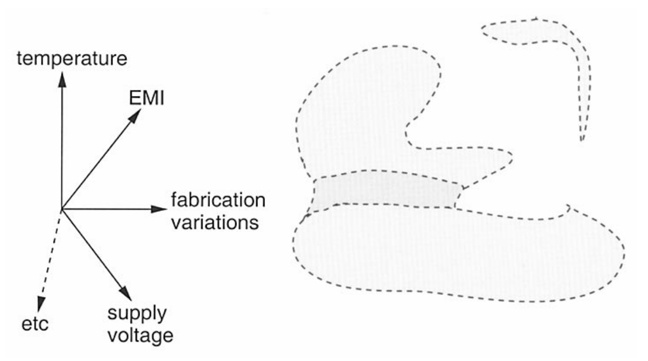


Fig. 1. Visualizing an operational envelope.

Not included above is the need to minimize the time and cost of the design process. At first sight this is not a requirement on the circuit itself, but in fact if the circuit is designed so as to be rapidly and rigorously testable and able to be easily modified to fix design or specification errors, then time to market is reduced. Additionally, a product lifecycle often includes the release of updates, or specially customized versions of the original design, requiring a readily modifiable design. Finally, engineering design is often a process of "design exploration" [9] in which partial attempts to solve the problem are used iteratively to refine the requirements specification into a useful form.

Hence a circuit must be evaluated according to many different criteria. For example, there can be as many as 20 different criteria composing the requirements for a commercial operational amplifier [10], [11]. Typically, some of the criteria should be maximized or minimized, while others provide hard constraints. EA methods for multicriteria [12] and constrained optimization [13] are well developed and there is potential for them to be adapted and extended for electronics design [14]–[20].

C. Unconstrained Evolutionary Electronics

In this paper, the "unconstrained" approach to evolutionary electronics is developed. We deliberately seek to explore beyond the scope of conventional design in the hope of finding circuits that are in some way superior in a practical setting, according to some combination of the many criteria above. This was stated as hypotheses H1–H3, where what is meant by "in some sense better" has now been clarified.

The need to reduce design time and cost means that electronics design automation through EA's could be profitable even if constrained to work within the same sphere as conventional design. By concentrating on the unconstrained case, however, we aim to map out the boundaries and benefits of the evolutionary approach, without blindly accepting the constraints inherited from conventional design. It is not claimed that all unconventional evolved circuits will be practically useful; it is claimed that to reject all unconventional circuits solely because they cannot be arrived at through conventional design methods is premature.

The argument that conventional design is constrained in a way that evolutionary design need not be, in principle, is

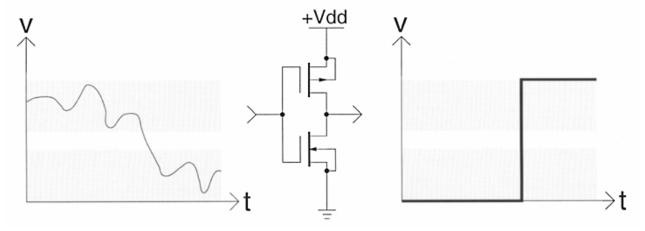


Fig. 2. Voltage/time waveforms exemplifying the operation of a CMOS NOT gate (center). Shown at the left is the input, and at the right is the corresponding output. The gray regions represent the basins of attraction of the two attractor states.

given in the next section. This is hypothesis H1. To show that this can allow a practical EA to find new useful circuits is an empirical matter (H2). Hence a pair of case studies is presented. Finally, research tools are described that are being used to illuminate whether unconventional evolved circuits really can be of engineering benefit, when judged by the many criteria of a practical application (H3).

II. Characterizing Conventional Design

First, we consider separately the two electronics paradigms: digital and analogue. Then the process of design within those paradigms is discussed.

A. Digital Design

In a digital design, whenever a signal is used or operated upon, it is rounded off to the closest one of a finite number of attractor states. Usually, there are two attractor states situated at the upper and lower extremes of possible signal levels: binary logic. Whenever a new or manipulated signal is generated, it is at one of the attractor states. Fig. 2 shows the operation of a binary NOT gate made from CMOS transistors.

It takes a nonzero time for an input to change from one state to another. During this time, and briefly afterwards, the fundamentally analogue nature of the transistors implementing a logic element is revealed at its output, which is to be seen briefly at levels *between* attractor states, possibly even making multiple transitions before quickly settling into the appropriate attractor state (switching transients).

In a digital circuit, made as a network of logic elements, precautions must be taken to ensure that switching transients do not affect the overall behavior of the system. This is done by some method of phase control between parts of the system, so that a subcircuit is not allowed to use the output of another until it has settled at the correct attractor state. In asynchronous logic design, local handshaking signals are used between subcircuits [21]. In synchronous logic, one or more regular "clock" signals is used to control when subcircuits can communicate. The frequency of a clock is chosen such that the slowest subcircuit it controls has time for its output to settle at the correct state between "ticks."

There are two main advantages to digital design. The first, to be elaborated in Section II-C, is that most of the design can be considered at the level of logical algebra, without having to think about the physics of electronic devices. The second is that digital systems, especially binary ones, are extremely tolerant to corruption of the signals. This is because of the signal restoration to an attractor state at almost every opportunity. It is relatively easy to make binary digital circuits with a huge operational envelope. For example, if a signal is corrupted by noise, or an attractor state of an element's output drifts with temperature, there are large margins for error, as seen in Fig. 2. In synchronous circuits, there also needs to be a margin for error included in the clock period: the time for an output to be computed, stabilize, and be transferred to the next subcircuit may vary within the operational envelope.

These advantages come at a cost. The basic elements must all have the signal-restoring property. The system must be broken down into subcircuits that are simple enough for internal transients not to be a problem. Usually this means that the subcircuits are basically feedforward or "combinatorial" (having no feedback or recurrent loops within them). The communications between the subcircuits must be carefully regulated, usually by means of registers that only load in new values when communications are to be permitted (subcircuits have stabilized). Digital design, by using the components only as switching devices, must impose these design constraints to suppress the other aspects of the analogue, continuous-time reality.[1]

B. Analogue Design

Analogue design finds analogies between the physical behavior of groups of electronic components and the operations needed to construct the desired system. To allow this exploitation of the natural behaviors of groups of components, internal signals are mostly of continuous value, in contrast to digital design. Although not unprincipled, analogue design is often thought of as more of an "art" compared to digital design. Even many predominantly digital systems require some analogue circuitry at least to deal with the interface to the world.

Analogue design can extract more functionality from the components than can digital, because more of the components' behavior is put to use. Especially potent is the use of real time, rather than representing time as a computational variable like any other, as in digital systems [24], [25]. For some tasks (where "task" includes nonbehavioral requirements) analogue design is clearly superior. In other cases, the choice of analogue or digital design is also strongly influenced by the ease of the design process itself, which often favors digital.

The disadvantages of analogue design are the obverse of the advantages of digital design identified above. Although system-level design can be at an abstract level, analogue circuit design must necessarily consider properties of the physical components, and their interactions, in greater depth than for digital design. The second disadvantage is that stability and large noise margins are more difficult to guarantee.

Sarpeshkar [26] shows that as the complexity of an analogue system increases, it becomes essential for signal values to

[1] Although charge is quantized, and these charges often move through periodic atomic lattices having discrete energy levels, at the scale addressed by contemporary circuit design we may consider currents and voltages as continuous [22]. This could change in the future if meso/nano-scale electronics becomes practical; it might then be possible to exploit physical quantization to implement digital computations [23].

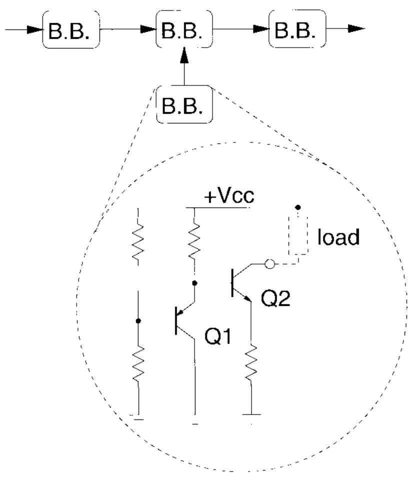


Fig. 3. Example of the building block (B.B.) approach in analogue design.

be restored to attractor states occasionally, otherwise all of the signal would eventually be swamped by unavoidable noise. This signal restoration does not happen at every circuit element, as it does in digital systems, and the representation of the analogue signal may be distributed across more than one physical voltage, current, or charge. There is flexibility in the choice of restoration schemes, not limited to two attractor states and not necessarily uniform throughout the system. Sarpeshkar provides a costs/benefits analysis for some restoration schemes under various resource constraints and operational envelopes, with reference to neural systems; it appears that there is great untapped potential in analogue (or hybrid) electronics, if new design styles could be developed.

In practice, analogue design is currently based around the use of building block subcircuits. These have been carefully designed and analyzed in the past to perform generally useful functions and form a "cookbook" with which a larger design can be constructed. A designer takes building blocks from textbooks and from looking at other circuits designed for similar problems. Examples of building blocks are amplifiers, filters, oscillators, voltage or current sources, and many others. The original design of a building block can be very challenging, often beyond the capabilities of designers who later use them. Without building blocks, the design of complex analogue systems would be too difficult.

The use of building blocks is not only to avoid "reinventing the wheel" for each design. By compartmentalizing the circuit into functional packages with well-defined interfaces, it becomes easier to make the whole circuit robust across an operational envelope. Taking thermal stability as an example, Fig. 3 shows a hypothetical system composed of building blocks. The enlarged subcircuit is a standard current-sink building block [27]. Transistor Q2 serves solely to compensate for the thermal variation of the base-emitter voltage of Q1. Under certain well-analyzed conditions, this building block can be used to serve the function of a current-sink over a range of temperatures, irrespective of what the rest of the system is doing. If all of the other building blocks are similarly endowed with thermal stability of function and are composed so as not to violate their constraints of operation, then the whole circuit can be given thermal stability. Although there are still pitfalls, the design of complex robust analogue circuits would be practically infeasible without this approach.

C. The Design Activity

We have seen that both digital and analogue design styles are, in practice, restricted as to what kinds of circuits can be produced. Digital designs must be made of signal-restoring elements regimented into subcircuits of simple dynamics, with constrained moments of communication. Analogue designs are made mostly of standard building blocks, with the rare creation of a completely novel design usually being restricted by design-difficulty to the level of these small subcircuits.

Further practical design constraints arise from the choice of design flow. A design flow starts with high-level design decisions about the gross structuring of the system and ends with exact details of how it should be implemented. In between are stages of problem decomposition and a progression from considering the problem at a highly abstract level, through to reifying the design to the concrete details of implementation. The particulars of a design flow are strongly influenced by the computer-aided design (CAD) tools available for each step, and the ease with which different tools can be integrated.

Design-automation tools are highly developed for digital systems, and far less so for analogue. Rutenbar [28] sees the impediment to analogue design automation as the difficulty of providing a library of standard cells adequate to implement an arbitrary user design fully. A standard cell is an implementation building block, completely specifying component details, placement, and routing to achieve a precise function: this is different from the more generic architectural design building blocks mentioned above. We have already indicated that it is inherent in an analogue design process that it is more difficult to abstract function from implementation, than for digital systems.

Evolutionary algorithms have been applied as design-automation tools at various levels of abstraction: Table I gives some examples for a digital design flow. Top-down design of this sort is ubiquitous and almost indispensable for large systems, but does impose constraints of its own on what circuits can be produced. Abstraction implies the suppression of some details; design at an abstract level therefore requires constraint of those neglected details upon progression toward the concrete implementation.

For the current enterprise, we seek to explore design space as freely as possible, even if in doing so we can only work with relatively small circuits in practice. Once the potential of new territories in design space is ascertained, this knowledge can be fed back into the domain of top-down design, perhaps as additions to the designer's cookbook of subcircuit ideas. The variation operators, or the representation of the circuit on which they operate, may be designed to encourage modularity and repeated structures, aiding the evolutionary process in the production of large systems [34], [35]. For the present, we apply evolutionary algorithms in a bottom-up fashion, with the EA's variation operators manipulating the structure at the finest level of implementation detail available. Constraints

TABLE I
SOME STEPS IN A DIGITAL DESIGN FLOW TO WHICH EA'S HAVE BEEN APPLIED

Design Activity	EA Example	Level
Partitioning into hardware and software modules	[29]	**Abstract**
Design of a network of high-level functions	[30]	$\Downarrow$
Function behaviour $\longmapsto$ Net of logic gates	[31]	$\Downarrow$
Net of logic gates $\longmapsto$ Choice from a library of physical components	[19]	$\Downarrow$
Net of physical components $\longmapsto$ Placed and routed VLSI layout	[32] [33]	**Concrete** (implementation)

associated with the conventional digital or analogue paradigms are resisted as being prejudicial for evolution. In general, the circuits are continuous-time, continuous-value dynamical systems; digital, analogue, and hybrid circuits are all included in this repertoire of possibilities. No claim is made that *more* of design space can practically be surveyed (that may or may not be the case), but rather new regions.

Here, there is no distinction between design and implementation: the process of evolutionary design happens at the implementation level. As well as avoiding abstraction constraints, this facilitates the integration of nonbehavioral and behavioral requirements. Many nonbehavioral requirements, such as size or power consumption, are closely coupled both with general design decisions and with details of the implementation. In a top-down design flow, implementation details are not fully contemplated during the early—more abstract—stages, making design for nonbehavioral requirements problematic. An example of integrating a fault-tolerance (nonbehavioral) requirement with embedded behavioral requirements will be given in Section III-C. It is partly the ability to embrace nonbehavioral requirements during all stages of an evolutionary design process, in combination with an exploration of new circuit structures and dynamics, that provides the opportunity for better circuits to arise through evolution (H3).[2]

By diagnosing the constraints of conventional design—from the analogue and digital design paradigms, and from top-down design flows—hypothesis H1 is shown. It is also deduced that an evolutionary approach, in principle, can explore some regions in design space that are beyond the scope of conventional methods. This is not sufficient for hypothesis H2: we need to go on to *demonstrate* an evolutionary algorithm exploring new regions. That is done through a pair of case studies carefully formulated for the purpose, presented in the next two sections. During these case studies, the need to deal with a realistic set of requirements is temporarily neglected. In the final section, ongoing work to address this is described, moving toward hypothesis H3: the practical evolution of robust unconventional electronics.

[2]It is not essential to adopt such a radical unconventional stance to begin to tackle this issue. For instance, Miller and Thomson [36] incorporate geometric layout considerations into evolutionary design of digital logic.

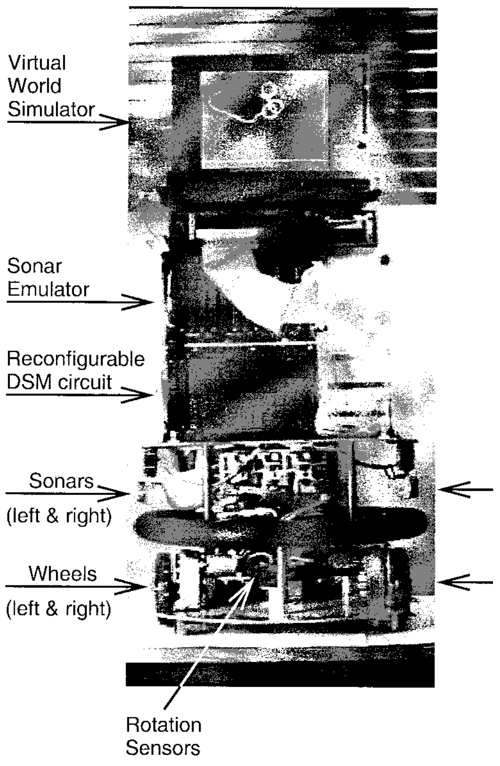


Fig. 4. The robot known as Mr Chips.

III. CASE STUDY 1: REMOVING DYNAMICAL CONSTRAINTS FROM A STANDARD ARCHITECTURE

In this first case study, a standard electronic architecture is taken, and the temporal constraints associated with digital design are relaxed and placed under evolutionary control. This allows a direct comparison of the behavioral capabilities of the same hardware when subjected to, or freed from, design constraints. Integration of a nonbehavioral requirement for fault tolerance with the behavioral requirements is then discussed. The behavioral requirements are an example of the embedded type, where it is the performance of an electromechanical system in which the circuit is embedded that is evaluated.

A. The Experiment

The circuit to be evolved was the onboard controller for the robot shown in Fig. 4 [37]. This two-wheeled autonomous mobile robot has a diameter of 46 cm, a height of 63 cm, and was required to display simple wall-avoiding/room-centering behavior in an empty 2.9 m $\times$ 4.2 m rectangular arena. For this scenario, the dc motors were not allowed to run in reverse and the robot's only sensors were a pair of time-of-flight sonars rigidly mounted on the robot, one pointing left and the other right. The sonars fire simultaneously five times a second; when a sonar fires, its output changes from logic $\mathbf{0}$ to logic $\mathbf{1}$, returning to $\mathbf{0}$ when the first echo is sensed at its transducer.

Conventional electronic design would tackle the control problem along the following lines: for each sonar, a timer would measure the length of its output pulses—and thus the time of flight of the sound—giving an indication of the range to the nearest object on that side of the robot. These timers would provide binary-coded representations of the two times of flight to a central controller. The central controller would be a hardware implementation of a finite-state machine (FSM), with the next-state and output functions designed so that it computes a binary representation of the appropriate motor speed for each wheel. For each wheel, a pulse-width modulator would take the binary representation of motor speed from the central controller and vary the mark : space ratio of pulses sent to the motor accordingly.

It would be possible to evolve the central controller FSM by implementing the next-state and output functions as lookup tables held in an off-the-shelf random access memory (RAM) chip; this is the well-known direct addressed ROM implementation of an FSM [38]. The FSM would then be specified by the bits held in the RAM, which could be reconfigured under the control of each individual's genotype in turn. Such evolution would still be subject to the constraints of digital design: all of the signals are synchronized to a global clock to give clean, deterministic state-transition behavior as predicted by an abstracted Boolean model.

What if the constraint of synchronization of all signals is relaxed and placed under evolutionary control? Although superficially similar to the FSM implementation, the result (Fig. 5) is a machine of a different nature. Not only is the global clock frequency placed under evolutionary control, but the choice of whether each signal is synchronized (latched) by the clock or whether it is asynchronous (directly passed through as an analogue voltage) is also determined by evolution. These relaxations of temporal constraints—constraints necessary for a designer's abstraction but not for unconstrained evolution—offer a rich range of potential dynamical behavior to the system, to the extent that the sonar echo pulses can be fed directly in, and the motors driven directly by the outputs, without any pre- or postprocessing: no timers or pulse-width modulators. (The sonar firing cycle is asynchronous to the evolved clock.)

Let this new architecture be called a dynamic state machine (DSM). It is not a finite-state machine because a description of its state must include the temporal relationship between the asynchronous signals, which is a real-valued analogue quantity. In the conventionally designed control system there was a clear sensory/control/motor decomposition (timers/controller/pulse-width-modulators), communicating in atemporal binary representations which hid the real-time dynamics of the sensorimotor systems, and the environment

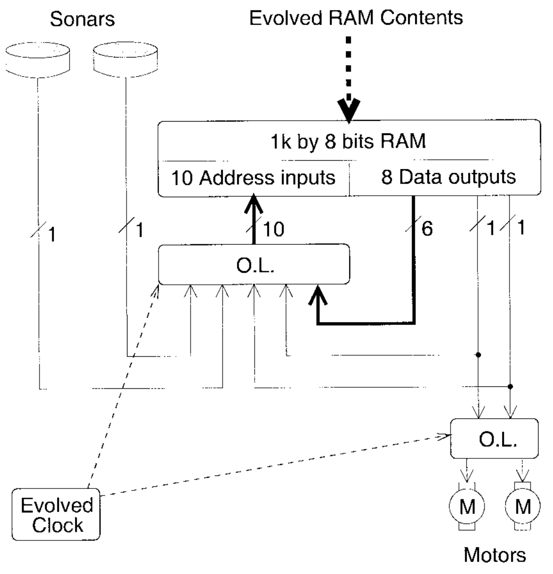


Fig. 5. The hardware implementation of the evolvable DSM robot controller. Optional latches (O.L.) control whether each signal is passed straight through asynchronously as an analogue voltage, or whether its digital value is latched according to the global clock of evolved frequency. Each optional latch was implemented using an analogue switch chip able to bypass a clocked latch. A full circuit diagram is given in [39].

linking them, from the central controller. Now, the evolving DSM is intimately coupled to the real-time dynamics of its sensorimotor environment, so that real-valued time can play an important role throughout the system. The evolving DSM can explore special-purpose tight sensorimotor couplings because the temporal signals can quickly flow through the system being influenced by, and in turn perturbing, the DSM on their way.

For the simple wall-avoidance behavior, only the outer two of the eight feedback paths seen in Fig. 5 were enabled, feeding the RAM chip's two least-significant data bits back to its two least-significant address inputs. The resulting DSM can be viewed as the fully connected, recurrent, mixed synchronous/asynchronous logic network shown in Fig. 6, where the bits stored in the RAM give a lookup table implementing any pair of logic functions of four inputs. This continuous-time dynamical system could not be simulated in software easily, because the effects of the asynchronous variables and their interaction with the clocked ones depend upon the characteristics of the hardware: meta stability [40], [41] and glitches will be rife, and the behavior will depend upon physical (analogue) properties of the implementation, such as propagation delays, meta-stability constants, and the behavior of the RAM chip when connected in this unusual way. Similarly, a designer would only be able to work within a small subset of the possible DSM configurations—the ones that are easier to analyze.

A simple GA was used, with a linear bit-string genotype, point-mutations, single-point crossover, linear rank-based selection, and elitism [42]. The contents of the RAM (only 32 bits required for the machine with two feedback paths), the period of the clock (16 bits in a Gray code, giving a clock frequency from around 2 Hz to several kHz) and the clocked/unclocked condition of each signal (1 bit per

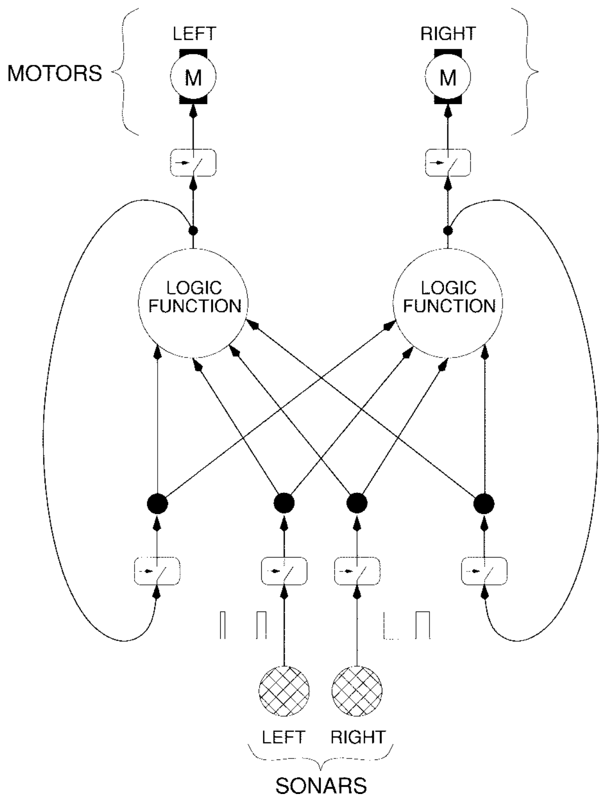


Fig. 6. An alternative representation of the evolvable DSM, as used in the experiment. Each [optional latch symbol] is an optional latch (see Fig. 5).

signal) were directly represented as contiguous segments of the genotype. The population size was 30, probability of crossover 0.7 per offspring, and the bit-wise mutation probability was set such that the expected number of mutations per offspring was one [43].

If the distance of the robot from the center of the arena in the x and y directions at time t was $c_x(t)$ and $c_y(t)$, then after an evaluation for T seconds, the robot's fitness was a discrete approximation to the integral

$$\text{fitness} = \frac{1}{T}\int_0^T \left(e^{-k_x c_x(t)^2} + e^{-k_y c_y(t)^2} - s(t)\right) dt. \quad (1)$$

k_x and k_y were chosen such that their respective Gaussian terms fell from their maximum values of 1.0 (when the robot was at the center of the arena) to a minimum of 0.1 when the robot was touching a wall in their respective directions. The function $s(t)$ encourages continual movement, having the value 0 when the robot is moving, but 1 otherwise. Each individual was evaluated for four trials of 30 s each, starting with different positions and orientations. The worst of the four scores was taken as the fitness [44]. For the final few generations, the evaluations were extended to 90 s, to find controllers that were not only good at moving away from walls, but also staying away from them.

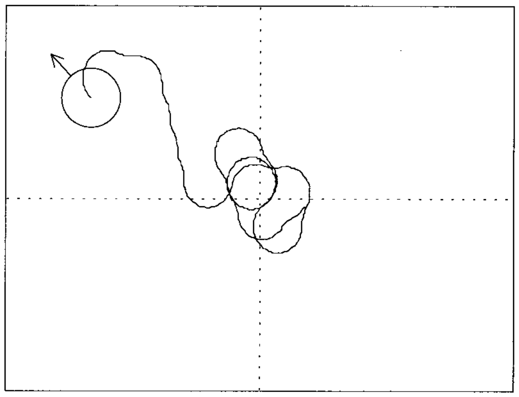
For convenience, evolution took place with the robot in a kind of "virtual reality." The real reconfigurable DSM circuit controlled the real motors, but the wheels were just spinning in the air. The photograph of Fig. 4 was taken during an actual evolutionary run of this kind. The wheels' angular velocities were measured and used by a real-time simulation of the motor characteristics to calculate how the robot would move if on the ground. The sonar echo signals were then artificially synthesized and supplied in real time to the hardware DSM. Realistic levels of noise were included in the sensor and motor models, both of which were constructed by fitting curves to experimental measurements, including a stochastic model for specular sonar reflections. Full details are given in [39]. The GA and the virtual environment simulation were performed by a laptop PC onboard the robot, and a pair of microcontrollers synthesized the sonar and clock waveforms. The real DSM hardware connected to the real motors was used at all times. For operation in the real world, the real sonars were simply connected in place of the simulated ones, and the robot placed on the ground.

Fig. 7 shows the excellent performance attained after 35 generations, with a good transfer from the virtual environment to the real world. The robot is drawn to scale at its starting position with its initial heading indicated by the arrow; thereafter only the trajectory of the center of the robot is drawn. The bottom-right picture is a photograph of behavior in the real world, taken by double-exposing 1) a picture of the robot at its starting position with 2) a long exposure of a light fixed on top of the robot moving in the darkened arena. If started repeatedly from the same position in the real world, the robot follows a different trajectory each time (occasionally very different) because of real-world noise. The robot displays the same qualitative range of behaviors in the virtual world, and the bottom pictures of Fig. 7 were deliberately chosen to illustrate this.

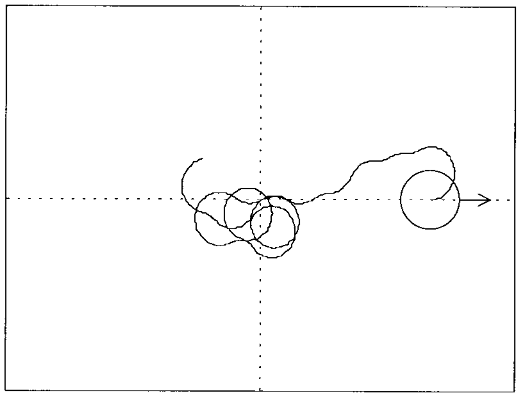
Given that this miniscule electronic circuit receives the raw echo signals from the sonars and directly drives the motors (one of which happens to be more powerful than the other), the performance is surprisingly good. It is not possible for the DSM directly to drive the motors from the sonar inputs (in the manner of Braitenberg's "Vehicle 2" [45]), because the sonar pulses are too short to provide enough torque. Additionally, such naive strategies would fail in the symmetrical situations seen Fig. 7(a) and (b). One of the evolved wall-avoiding DSM's was analyzed (below), and found to be going from sonar echo signals to motor pulses using only 32 bits of RAM and 3 flip-flops (excluding clock generation): highly efficient use of hardware resources, made possible by the absence of design constraints.

B. Analysis

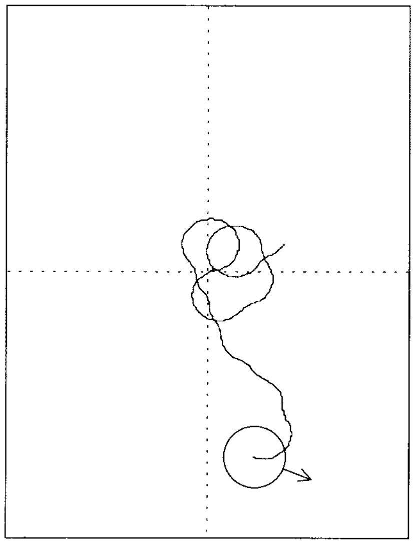
Fig. 8 attempts to represent one of the wall-avoiders in state-transition format. This particular individual used an evolved clock frequency of 9 Hz (about twice the sonar pulse repetition rate). Both sonar inputs evolved to be asynchronous, and both motor outputs clocked, but the internal state variable that was clocked to become the left motor output was free-running (asynchronous), whereas that which became the right output was clocked. In the diagram, the dotted state transitions occur as soon as their input combination is present, but the solid transitions only happen when their input combinations are



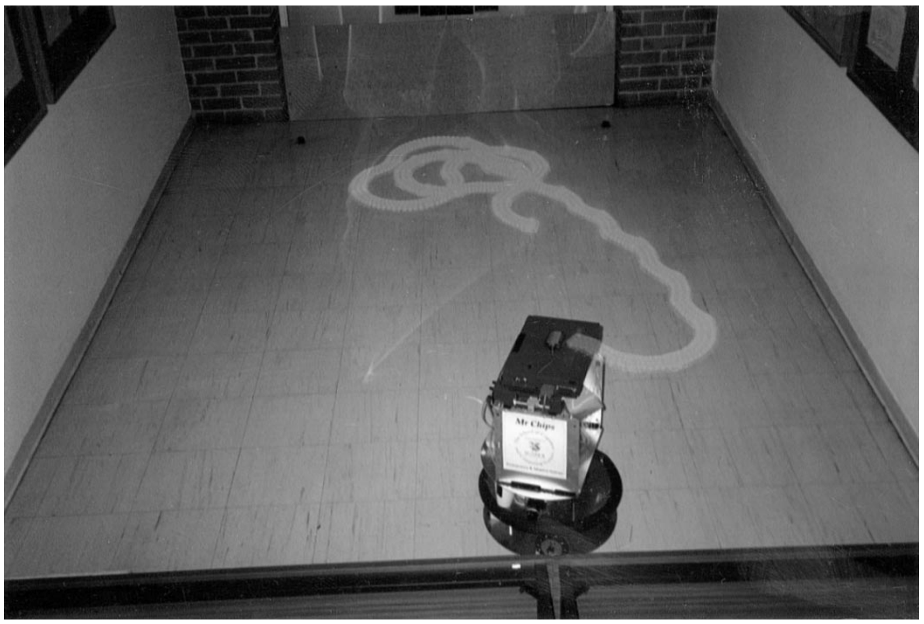
(a)



(b)



(c)



(d)

Fig. 7. Wall avoidance in virtual reality and (d) in the real world, after 35 generations. The top pictures are of 90 s of behavior, the bottom ones of 60 s.

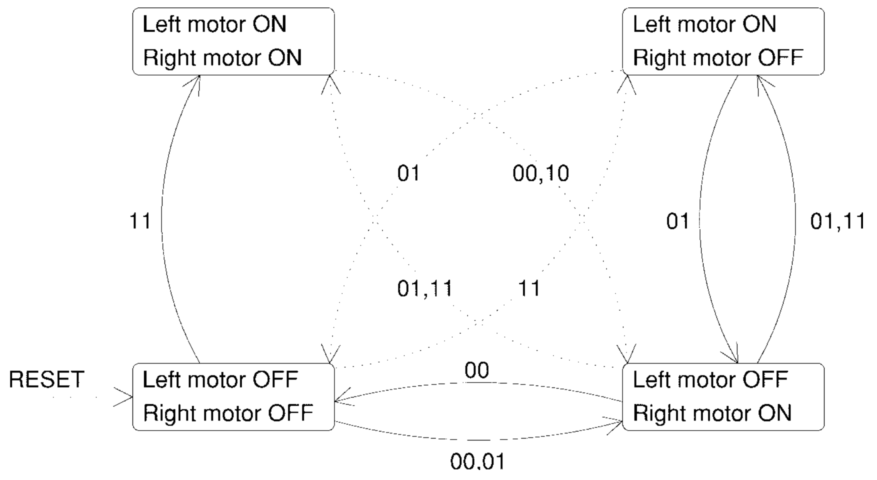


Fig. 8. A representation of one of the wall-avoiding DSM's. Asynchronous transitions are shown dotted and synchronous transitions solid. The transitions are labeled with (left, right) sonar input combinations, and those causing no change of state are not shown. There is more to the behavior than is seen immediately in this state-transition diagram, because it is not entirely a discrete-time system, and its dynamics are tightly coupled to those of the sonars and the rest of the environment.

present at the same time as a rising clock edge. Since both motor outputs are synchronous, the state can be thought of as being sampled by the clock to become the motor outputs.

This state-transition representation is misleadingly simple in appearance, because when this DSM is coupled to the input waveforms from the sonars and its environment, its dynamics are subtle, and the strategy being used is not obvious. It is possible to convince oneself that the diagram is consistent with the behavior, but it would have been very difficult to predict the behavior from the diagram because of the rich feedback through the environment and sensorimotor systems on which this machine relies. The behavior even involves a stochastic component, arising from the probabilities of certain combinations of the machine's mixed synchronous/asynchronous state, at the arrival of pulses from the clock and the asynchronous sonars.

The DSM underlies a nontrivial robot behavior, using minimal resources [46], by means of the circuit's rich dynamics and exploitation of the hardware (an idea dating back to 1949 [47]). After relaxing the temporal constraints necessary to support the designers' digital abstraction, a tiny amount of hardware has been able to display rather surprising abilities. As a control experiment, three GA runs were performed under identical conditions, but with all of the optional latches set to "clocked" irrespective of the genotype. All three runs failed completely, confirming that new capabilities had been released from the architecture when the dynamical constraints were relaxed. In another set of three control runs, all the optional latches were set to "unclocked." These runs succeeded but the behavior was not so reliable: from time to time the robot would head straight for a wall and crash into it.

In three repetitions of the main experiment, the clock allowed the mixed synchronous/asynchronous controllers to move with a slight "waggle" [just visible in the Fig. 7(d)], and this prevented them from being disastrously fooled by specular sonar reflections. The sonars were effectively scanning the walls slightly because of the waggling movement of the robot body. This suggests that while removing an enforced clock can widen the repertoire of dynamical behaviors, providing an optional clock of evolvable frequency, to be used at points in the circuit determined by evolution, can expand the repertoire of dynamics still further. The clock becomes a resource, not a dynamical constraint.

C. Integrating a Nonbehavioral Requirement of Fault Tolerance

In Section II-C we described the worth of integrating nonbehavioral requirements into the objectives during evolutionary design. Some nonbehavioral characteristics, such as size or power consumption, are usually easily measurable to become factors in the selection process [14]. Others can be impractical to measure directly, but sometimes the evaluation procedure can be contrived so that they exert an implicit influence on fitness. As an example, we consider introducing a fault tolerance requirement for the DSM robot controller. Evolutionary techniques for fault-tolerant electronics are largely unexplored, but the engineering benefits on offer are significant [48], [49], [6].

The requirement is for the controller to be tolerant to any adverse single-stuck-at (SSA) fault in the memory array of the RAM chip, causing a bit to read the opposite of the value written to it. These faults are easily emulated by inverting one of the bits specified by evolution as it is written to the chip. Altering the configuration is a generally applicable way to emulate some classes of faults in reconfigurable hardware; if the circuits are being evaluated in software simulations there is even more flexibility.

Ideally, for each fitness evaluation an individual would be given a trial in the presence of every possible fault in turn, and the resulting fitness score would be some measure of performance in the face of any fault. There are usually many possible faults, however, making exhaustive testing prohibitively time consuming (but not always [50]). Even for the DSM robot controller, testing each individual in the presence of the 32 emulated adverse SSA faults would take too long.

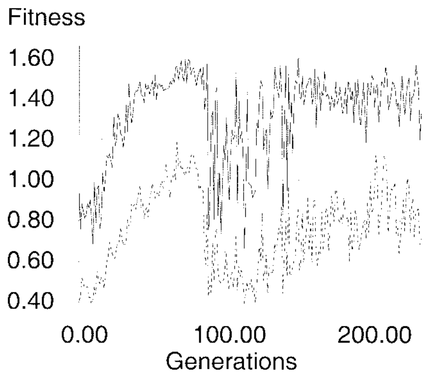


Fig. 9. Maximum and mean fitness in the population over time. The first 85 generations were in the absence of faults, thereafter all fitness evaluations were in the presence of the current fault.

If the goal is to optimize worst-case performance (i.e., minimize the effects of the most serious fault), there is a potential shortcut. In this case the fitness measure will be based on performance in the presence of only the single most serious fault. If some way of predicting which fault is the most serious can be found, then only this single fault needs to be introduced during the fitness evaluation. A similar situation arises if only a relatively small subset of the possible faults seriously degrades the system: only this subset need be considered.

Which faults are the most serious, however, might be different for each individual in the population. If the only way to identify the worst faults for each individual is to test them with each fault in turn, there can be no shortcut. In practice, though, after the first few generations the individuals are mostly similar and the population as a whole changes gradually over time. These facts can be used in predicting which faults are the most serious without having to test every individual with every fault.

We now apply these ideas to the robot controller. First, the wall-avoider was evolved as before, but this time using a population size of 50. After 85 generations the GA had stabilized at a good solution. Then the *consensus sequence* was generated: the genotype formed by, for each locus, taking whichever of the values {0, 1} was most common in the population at that position. The robot controller coded for by this consensus sequence was then tested in the presence of each of the 32 possible adverse SSA faults in turn. The fault that caused the consensus individual to behave the most poorly (lowest fitness score) was nominated as the "current fault." Another generation of evolution was then performed, but with the current fault being present during all of the fitness evaluations. After this generation the new consensus individual was constructed, tested, and a possibly new current fault nominated for the next generation. The process continued in this way, with a single fault being present throughout all evaluations within a generation—this fault being the one that caused the worst performance in the consensus individual of the previous generation.

Fig. 9 shows that the maximum and mean fitnesses dropped sharply at generation 85 when faults were first introduced, but over the course of the next 150 generations returned to high values. Fig. 10 shows that when the faults were first applied

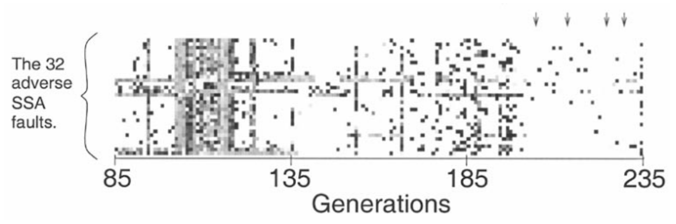


Fig. 10. The evolution of fault tolerance: results of the exhaustive test over all possible adverse SSA faults made on the consensus individual of each generation. The darker a spot, the more serious the fault. Pure white represents satisfactory performance (fitness $\geq$ 1.0), and pure black the worst possible performance. At the generations marked with arrows, the consensus individual is satisfactory in the presence of any SSA fault.

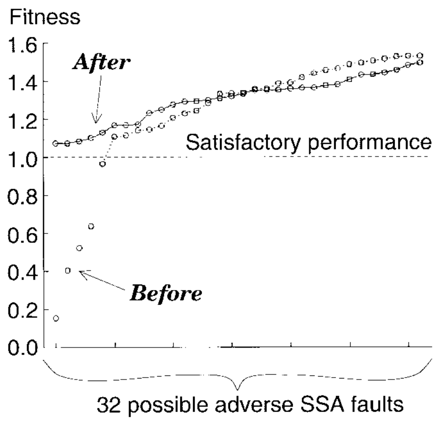


Fig. 11. Fault tolerance of the consensus at generation 85 and then after 119 generations of evolution in the presence of faults. In each case, the faults have been sorted from left to right in order of severity.

the controller was already tolerant to most SSA faults, but that a few were critical. At various stages afterwards, this tolerance to most SSA faults is lost in the GA's attempts to improve performance on the single most serious current fault. Some serious faults are seen to persist over long periods. Eventually, consensus individuals arose that give satisfactory performance when any of the SSA faults is present.[3] Fig. 11 compares the fault tolerance of the conventionally evolved consensus individual at generation 85 with that of the first completely tolerant consensus which arises at generation 204. The criterion for satisfactory performance was for the real robot to display what would reasonably be called wall-avoiding behavior and corresponds to a fitness score of $\geq$1.0.

This crude approach has exploited the similarity between individuals in the population by predicting that a single fault will be the most serious one for all individuals at a particular generation. This fault was identified by exhaustively testing a single "average" individual—the consensus. Though this fault-prediction strategy is not exact, it had the desired effect of catalyzing the evolution of a completely fault-tolerant individual.

[3] If the GA was left to run, then these completely tolerant solutions would be lost again as the GA concentrated entirely on improving performance in the presence of the current most serious fault—even if that performance was already satisfactory. No claims are made regarding the generality and reliability of this particular algorithm used here for illustrative purposes.

Many other strategies could be used to decide which faults an individual should encounter during its evaluation: the example above is merely a simple illustration. If there were many possible faults, exhaustive testing of even just the single consensus individual per generation could take too long. One suggestion is to co-evolve [51] a population of faults that concentrates on the weak-spots of the target population and tracks them over time [49], [6]. Another potential method is the use of repeated reevaluations of the more successful individuals to build up gradually an accurate picture of their performance in the presence of a set of faults [52].

Whatever method is used, it seems that if some way of targeting the most serious weak spots of individuals can be found, then subjecting the individuals to these faults during their fitness evaluations can cause the evolution of systems tolerant to all of the possible faults. It may be possible to use an adaptive process such as co-evolution to target the weak spots or search using application-specific heuristics may prove more appropriate.

D. Conclusion to Case Study 1

The DSM robot controller experiments showed unequivocally that removing the constraints of conventional design—in this case, the temporal constraints associated with the digital design paradigm—can release greater behavioral capabilities from essentially the same electronic components. They provided an example of evolution using this freedom to explore beyond the scope of conventional design: hypotheses H1 and H2 are demonstrated.

A technique for the evolution of fault-tolerant circuits was presented. It is not clear that it can scale up to circuits with many possible faults, so it may be restricted to small circuits (or subcircuits), and to problems posed such that there are many satisfactory solutions. Nevertheless, by integrating the nonbehavioral requirement of fault tolerance with the behavioral requirements, a robot controller was evolved that was *inherently* fault tolerant, not relying on the explicit use of spare (redundant) parts as is normal [53], [54]. This was possible because the nonbehavioral, implementation-oriented, requirement was an inherent part of the evolutionary design process, and was not deferred to the late stages of a top-down design flow.

IV. Case Study 2: Evolving a Circuit with Minimal Constraints

In the previous case study, some temporal constraints were relaxed, but the general architecture of the system was fixed. The next step is to discover whether evolution really can produce circuits looking completely alien to an electronics designer or whether in practice such bizarre circuits are unworkable. As first moves toward this radical goal, the evolution of unusual oscillator circuits was investigated, both in simulation [37] and using reconfigurable chips [55]. The latter has been greatly extended and studied rigorously by Huelsbergen *et al.* [56]. We now elucidate further by studying another task.

The electronic device selected for the experiments is reconfigurable at a very fine grain, so as to impose the minimum of architectural constraints: the Xilinx XC6216 field-programmable gate array (FPGA) [57]. Fig. 12 shows the subset of its functionality used in the experiment. There is a 64 $\times$ 64 array of cells on the chip, of which only the north-west 10 $\times$ 10 corner was used. The connections between cells, and their internal functions, are controlled by multiplexers. These multiplexers are configured according to the contents of RAM distributed throughout the array. A host microprocessor can write to this configuration memory, causing the multiplexers (electronic switches) to be set in a particular way, physically instantiating any one of a vast number of possible electronic circuits on the chip. That circuit will then behave in real time, according to semiconductor physics, without any further intervention. Special blocks around the periphery of the array interface the edge cells to the external pins of the chip and can be configured as inputs or outputs. In the experiment, there is one input and one output, configured at fixed positions chosen by the investigators.

The function F within a cell can be configured to be any binary logic function of two inputs, or multiplexer functions of three inputs. In the experiment, however, the design constraints needed for digital operation will not be imposed. The circuit being evolved is a continuous-time, continuous-valued, dynamical system. The components of this system (the cell function and routing multiplexers) have a very high gain, so that if digital design principles are followed then the signals will nearly always be saturated fully high or low. Without these design constraints, there is also the possibility for analogue effects. For example, a NOT gate is physically a very high-gain inverting amplifier, and evolution is free to use it as such.

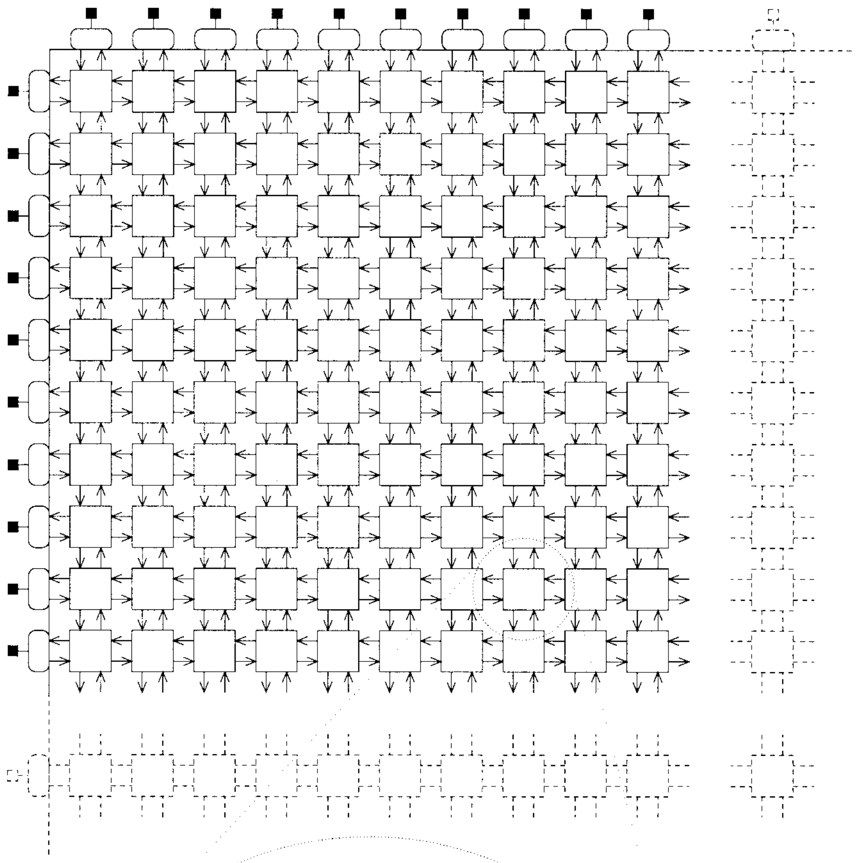
A. The Experiment

The task was to evolve a circuit—a configuration of a 10 $\times$ 10 corner of the XC6216 FPGA—to discriminate between square waves of 1 kHz and 10 kHz presented at the input [55]. Ideally, the output should go to +5 V as soon as one of the frequencies is present, and to 0 V for the other. The task was intended as a first step into the domains of pattern recognition and signal processing, as well as being part of the scientific program. One could imagine such a circuit being used to demodulate frequency-modulated binary data received over a telephone line.

This task was not facile, because few components were provided and the circuit has no access to a clock, or other off-chip resources such as RC time constants, by which the period of the input could be timed or filtered. Evolution was required to produce a configuration of the 100 cells to discriminate between input periods five orders of magnitude longer than the input $\Rightarrow$ output propagation time of each cell (which is just a few nanoseconds). A continuous-time recurrent arrangement of the 100 cells had to be found that could perform the task entirely on-chip.

Although the results of Section III suggested a benefit in providing a clock of evolvable frequency as an optional

(a)



(b)

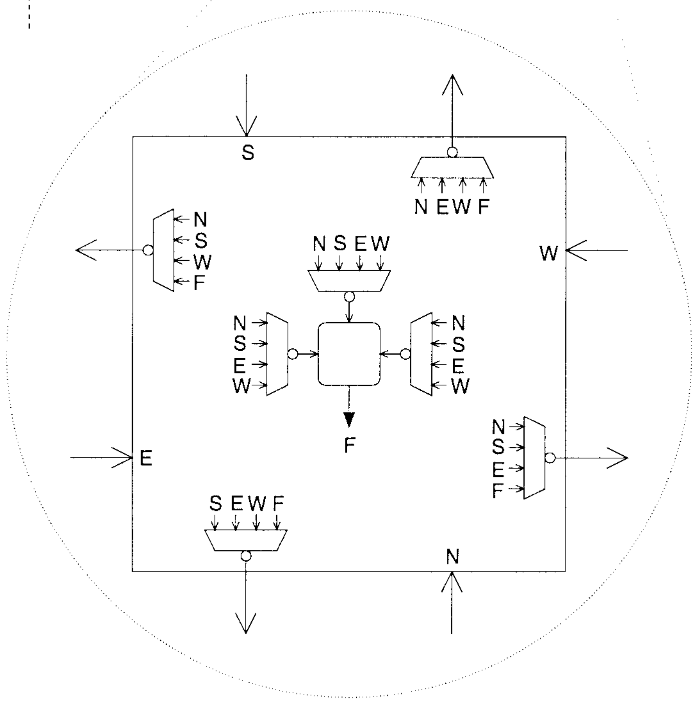


Fig. 12. A simplified view of the XC6216FPGA. Only those features used in the experiments are shown. (a) A 10 × 10 corner of the 64 × 64 array of cells. (b) The internals of an individual cell, showing the function unit at its center. The symbol represents a multiplexer—which of its four inputs is connected to the output (via an inversion) is controlled by the configuration memory. Similar multiplexers are used to implement the user-configurable function F.

resource rather than as an imposed constraint, no clock was made available. This was primarily to assess the possibility of evolution of very unusual circuits. There is also an engineering justification: the components needed for an external time reference would be bulky compared to the 1% of the FPGA's silicon area used by the final evolved circuit. The fully

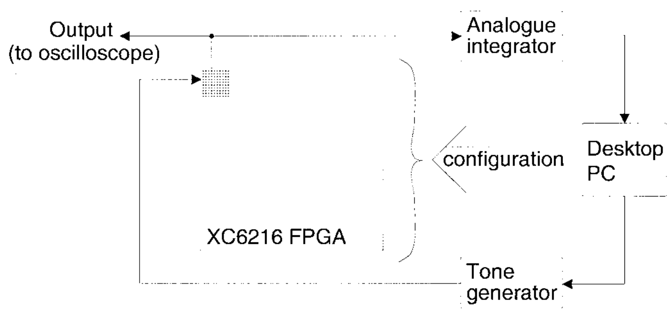


Fig. 13. The apparatus for the tone discriminator experiment. The 10 × 10 corner of cells used is shown to scale with respect to the whole FPGA. The single input to the circuit was applied as the east-going input to a particular cell on the west edge, as shown. The single output was designated to be the north-going output of a particular cell on the north edge.

integrated solution is preferable in terms of size, mechanical robustness, and the cost of components and manufacturing.

To configure a single cell, there were 18 multiplexer control bits to be set, and these bits were directly encoded onto a linear bit-string genotype. The genotype of length 1800 bits was formed from left to right by taking the cells in the 10 × 10 corner in a raster fashion, from west to east along each row, and taking the rows from south to north. A basic GA was again used, with a population size of 50, a crossover probability of 0.7, and a per-bit mutation probability such that the expected number of mutations per genotype was 2.7. The mutation rate was derived empirically.

The GA was run on a normal desktop PC interfaced to some simple in-house electronics[4] as shown in Figs. 13 and 14. To evaluate the fitness of an individual, the hardware-reset signal of the FPGA was first momentarily asserted to make certain that any internal conditions arising from previous evaluations were removed. Then the 1800 bits of the genotype were used to configure the 10 × 10 corner of the FPGA, and the FPGA was enabled. At this stage a genetically specified circuit existed on the chip, behaving in real-time according to semiconductor physics.

The fitness of a physically instantiated circuit was evaluated automatically as follows. A tone generator drove the circuit's input with five 500 ms bursts of the 1 kHz square-wave, and five of the 10 kHz wave. These ten test tones were shuffled into a random order, which was changed every time. There was no gap between the test tones. An analogue integrator was reset to zero at the beginning of each test tone and then it integrated the voltage of the circuit's output pin over the 500 ms duration of the tone.

Let the integrator reading at the end of test tone number t be denoted i_t $(t = 1, 2, \ldots, 10)$. Let S_1 be the set of five 1 kHz test tones and S_{10} the set of five 10 kHz test tones. Then the individual's fitness was calculated as

$$\text{fitness} = \frac{1}{5}\left|\left(k_1 \sum_{t \in S_1} i_t\right) - \left(k_2 \sum_{t \in S_{10}} i_t\right)\right|. \tag{2}$$

This fitness function demands the maximizing of the difference between the average output voltage when a 1 kHz input is present and the average output voltage when the 10 kHz input is present. The calibration constants k_1 and k_2 were determined empirically, such that circuits simply connecting their output directly to the input would receive zero fitness. Otherwise, with $k_1 = k_2 = 1.0$, small frequency-sensitive effects in the integration of the square-waves were found to make these useless circuits a troublesome local optimum. This fitness specification is an example of direct behavioral requirements.

It is important that the evaluation method—here embodied in the analogue integrator and the fitness function (2)—facilitates an evolutionary pathway of very small incremental improvements. Earlier attempts, where the evaluation method only paid attention to whether the output voltage was above or below the logic threshold, met with failure.

B. Results

Throughout the experiment, an oscilloscope was directly attached to the output pin of the FPGA (see Fig. 13), so that the behavior of the evolving circuits could be visually inspected. Fig. 15 shows photographs of the oscilloscope screen, illustrating the improving behavior of the best individual in the population at various times over the course of evolution.

The individual in the initial random population of 50 that happened to get the highest score produced a constant +5 V output at all times, irrespective of the input. It received a fitness of slightly above zero just because of noise. Thus, there was no individual in the initial population that demonstrated any ability whatsoever to perform the task.

For the first few hundred generations, the "best" circuits simply copied the input to the output, combined with various high-frequency oscillatory components. By generation 650, definite progress had been made. For the 1 kHz input, the output appeared to stay mostly high; for the 10 kHz input, the output resembled the input.

The photographs (Fig. 15) show the behavior gradually being refined, finally reaching the perfect desired behavior. In fact, not visible in the final photographs were infrequent unwanted spikes in the output; these were finally eliminated around generation 4100. The GA was run for a further 1000 generations without any observable change in the behavior of the best individual. The final circuit (arbitrarily taken to be the best individual of generation 5000) appears to be perfect when observed by eye on the oscilloscope. If the input is changed from 1 kHz to 10 kHz (or vice-versa), then the output changes cleanly between a steady +5 V and a steady 0 V without any perceptible delay.

It is apparent from the oscilloscope photographs that evolution explored beyond the scope of conventional design. For instance, the waveforms at generation 1400 would seem absurd to an electronics designer of either digital or analogue schools.

[4]*Technical Notes:* The circuitry was mounted wire-wrapped on an ISA (industry standard architecture) card (Fig. 14). The analogue integrator was of the basic op-amp/resistor/capacitor type, with a MOSFET to reset it to zero [27]. A MC68HC11A0 microcontroller operated this reset signal (and that of the FPGA), generated the tones, and performed 8-bit A/D conversion on the integrator output. A final accuracy of 16 bits in the integrator reading was obtained by summing (in software) the result of integration over 256 sub-intervals. Locations in the configuration memory of the FPGA and in the dual-port RAM used by the microcontroller could be read and written by the PC via registers mapped into the ISA-Bus I/O space. The XC6216 device was a preproduction engineering sample.

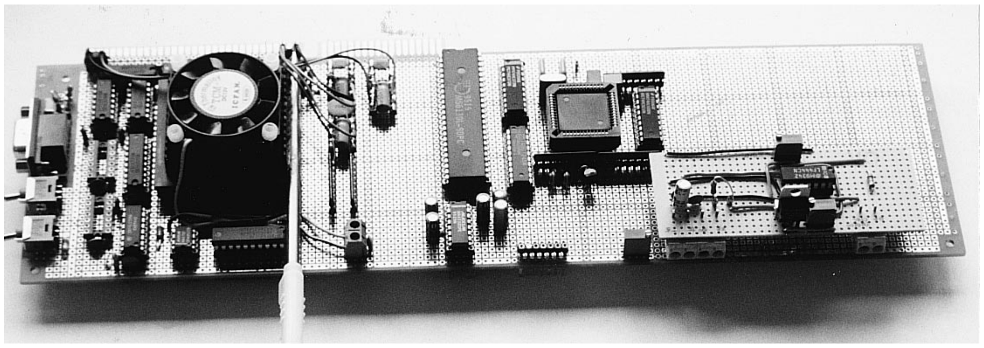


Fig. 14. The circuitry to evolve the tone discriminator. This ISA card plugs directly into the PC, and no extra electronics is needed. (a) The FPGA (beneath a fan-cooled heat sink) and its interface chips. (b) A microcontroller supervising fitness evaluations. (c) The analogue integrator.

Not so evident in these photographs is the rich range of dynamical time scales actually present. The components of the nominally digital FPGA were not used according to a binary logic abstraction, because a wider repertoire of behaviors was available in the absence of design constraints.

Graphs of maximum and mean fitness and of genetic convergence are given in Fig. 16. These graphs suggest interesting population dynamics, especially at around generation 2660. The experiment is analyzed in depth from an evolution-theoretic perspective in [58]. Crucial to any attempt to understand the evolutionary process that took place is the observation that the population had formed a genetically converged "species" before fitness began to increase: this is contrary to some conventional GA thinking [43], [44]. Evolution was the process of continual adaptation of this relatively converged species, with mutation playing a key role in generating new variants. Neutral networks—pathways of mutational change having no effect on fitness—are thought to have been important in allowing continued exploration without becoming stuck at poor local optima [58].

The entire experiment took two–three weeks. This time was dominated by the 5 s taken to evaluate each individual, with a small contribution from the process of calculating and saving data to aid later analysis. The times taken for the application of selection, the variation operators, and to configure the FPGA were all negligible in comparison. Current work suggests that the fitness tests could have used much shorter bursts of input tones. If evolution is to be free to exploit all of the components' physical properties, fitness evaluations must take place at the real time scales of the task to be performed and cannot simply be accelerated, as they could for a discrete-time system by increasing the clock speed. The evolution of circuits in detailed physical simulations is increasingly attractive as computer-power increases, but would be infeasible for circuits of this complexity in the near future. See Section V-C for the use of simulations.

The final circuit is shown in Fig. 17; observe the many continuous-time feedback paths. The lack of visible patterns in the circuit structure is not surprising: no preconceived bias toward modular or repeated substructures was applied, nor is it apparent that such patterning would be appropriate for such a small circuit and for this task.

Parts of the circuit that could not possibly affect the output can be pruned away. This was done by tracing all possible paths through the circuit that eventually connect to the output. A path not only includes routing, but also passing from an input to the output of a cell's function unit. It was assumed that all of a function unit's inputs could affect the function unit output, even when the nominal logic function indicated that this should not be so. This assumption was made because it was not known exactly how function units that were connected in continuous-time feedback loops actually would behave. In Fig. 18, cells and wires are only drawn if there is a connected path by which they could possibly affect the output, which leaves only about half of them.

The pruned diagram shows components that could be functional, but does not guarantee that they all are. To find which parts were actually contributing to the behavior, a search was conducted to find the largest set of cells that could have their function unit outputs simultaneously clamped to constant values (**0** or **1**) without affecting the behavior. To clamp a cell, the configuration was altered so that the function output of that cell was sourced by the flip-flop inside its function unit (a feature of the chip not mentioned until now, and which was not used during evolution): the contents of these flip-flops can be written by the PC and can be protected against any further changes. A program was written to select a cell at random, clamp it to a random value, perform a fitness evaluation, and to return the cell to its unclamped configuration if performance was degraded, otherwise to leave the clamp in place. This procedure was iterated, gradually building up a maximal set of cells that can be clamped without altering fitness. The order of clamping attempts, and the clamping values chosen, could affect the result; hence the whole exercise was repeated until a clear consensus emerged as to what the largest clamping set was.

In the above automatic search procedure, the fitness evaluations were more rigorous (longer) than those carried out during evolution, so that very small deteriorations in fitness would be detected (made difficult by the noise present during all evaluations). Clamping some of the cells in the extreme north-west corner produced so tiny a fitness decrement that even the extended evaluations did not detect it, yet by the time all of these cells of small influence had been clamped, the

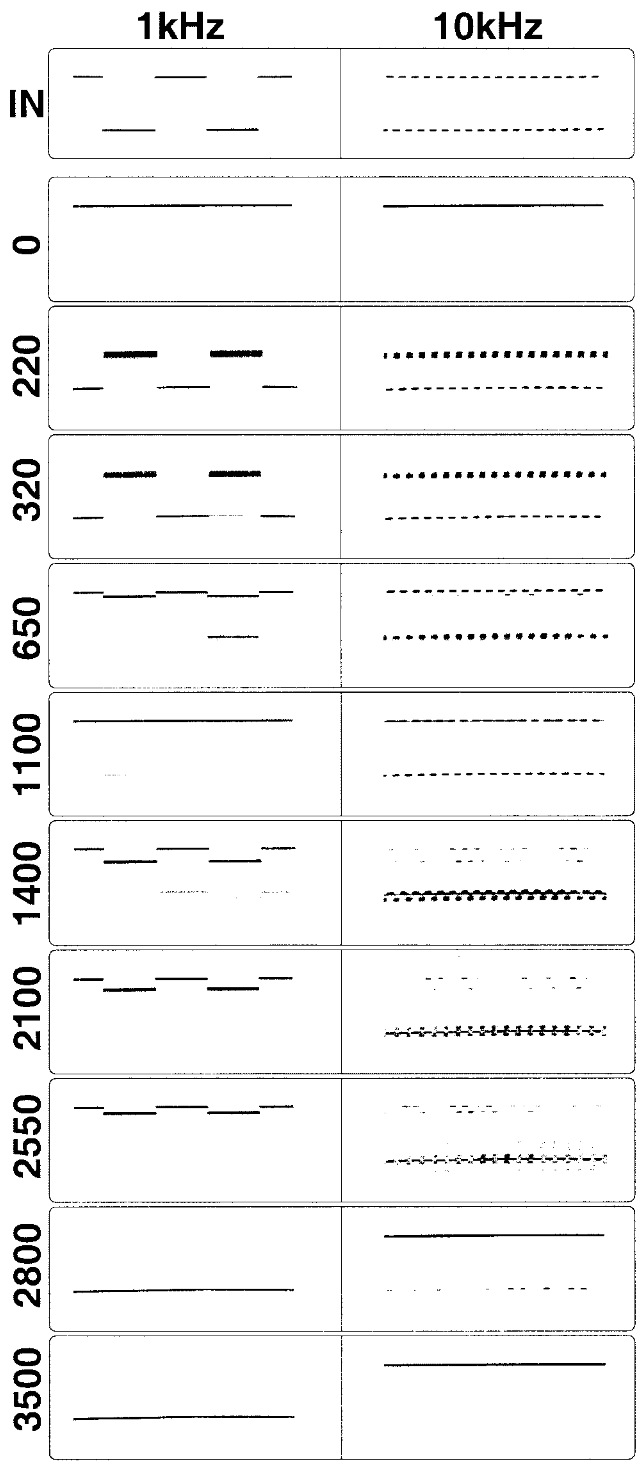


Fig. 15. Photographs of the oscilloscope screen. Top: The 1 kHz and 10 kHz input waveforms. Below: The corresponding output of the best individual in the population after the number of generations marked down the side. For these photographs an analogue oscilloscope of 20 MHz bandwidth was used.

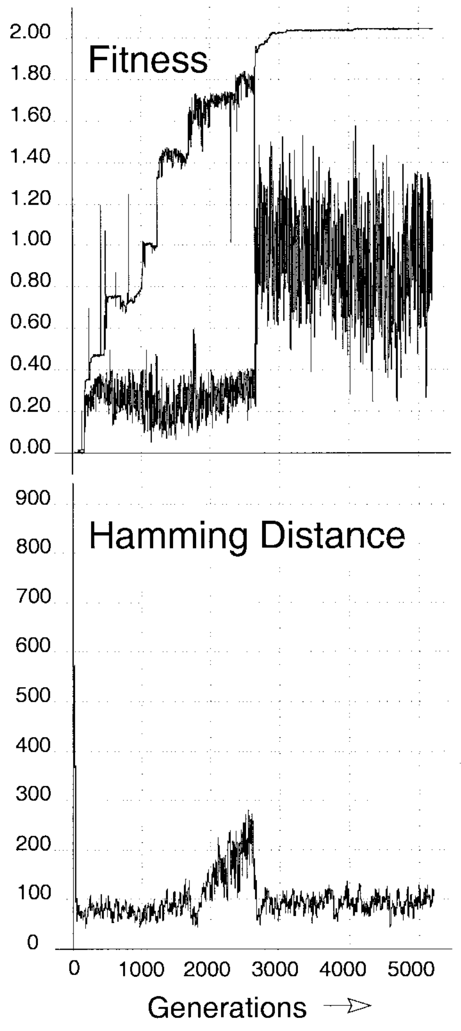


Fig. 16. Population statistics. Top: Maximum and mean fitnesses of the population at each generation. Below: Genetic convergence, measured as the mean Hamming distance between the genotypes of pairs of individuals, averaged over all possible pairs.

combined effect on fitness was quite noticeable. In these cases manual intervention was used (informed by several runs of the automatic method), with evaluations happening by watching the oscilloscope screen for several minutes to check for any infrequent spikes that might have been caused by the newly introduced clamp.

Fig. 19 shows the functional part of the circuit that remains when the largest possible set of cells has been clamped without affecting the behavior. The cells shaded gray cannot be clamped without degrading performance, even though there is no connected path by which they could influence the output—they were not present on the pruned diagram of Fig. 18. Clamping one of the gray cells in the north-west corner has only a small impact on behavior, introducing either unwanted pulses into the output, or a small time delay before the output changes state when the input frequency is changed. Clamping the function unit of the most south-eastern gray cell, which also has two active connections routed through it, degrades operation severely—even though that function output is not used as an input to any other cells. The gray cells must be influencing the rest of the circuit by some means other than the normal inter-cell routing; this enigma will be revisited in the analysis to follow.

This circuit is discriminating between inputs of period 1 ms and 0.1 ms using only 32 cells, each with a propagation delay of less than 5 ns, and with no off-chip components whatsoever:

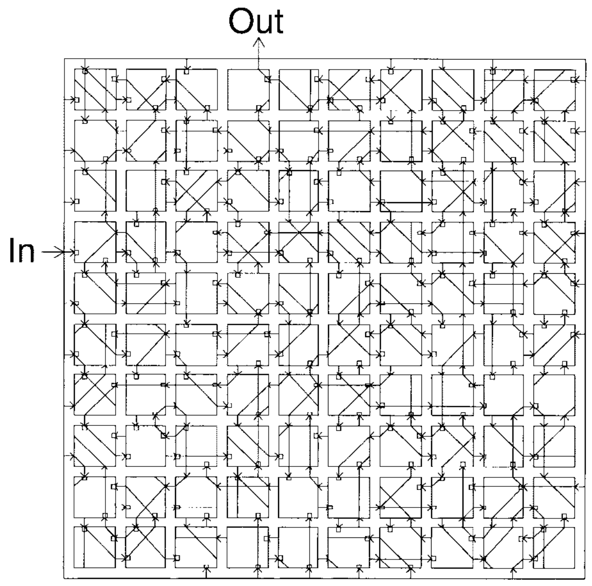


Fig. 17. The final evolved circuit. The 10 × 10 array of cells is shown, along with all connections that eventually connect an output to an input. Connections driven by a cell's function output are represented by arrows originating from the cell boundary. Connections into a cell which are selected as inputs to its function unit have a small square drawn on them. The actual setting of each function unit is not indicated in this diagram.

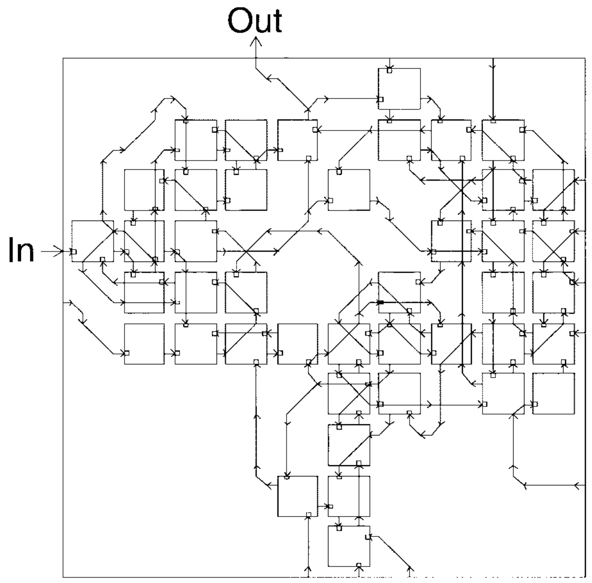


Fig. 18. The pruned circuit diagram: cells and wires are only drawn if there is a connected path through which they could possibly affect the output.

a surprising feat. Evolution has been free to explore the full repertoire of behaviors available from the silicon resources provided, even being able to exploit the subtle interactions between adjacent components that are not connected directly. The input/output behavior of the circuit is digital, because that is what maximizing the fitness function required, but the complex analogue waveforms seen at the output during the intermediate stages of evolution betray the rich continuous-time continuous-value dynamics that are likely to be present internally.

Only a core of 32 out of the 100 cells is involved in generating the behavior, even though there was no explicit

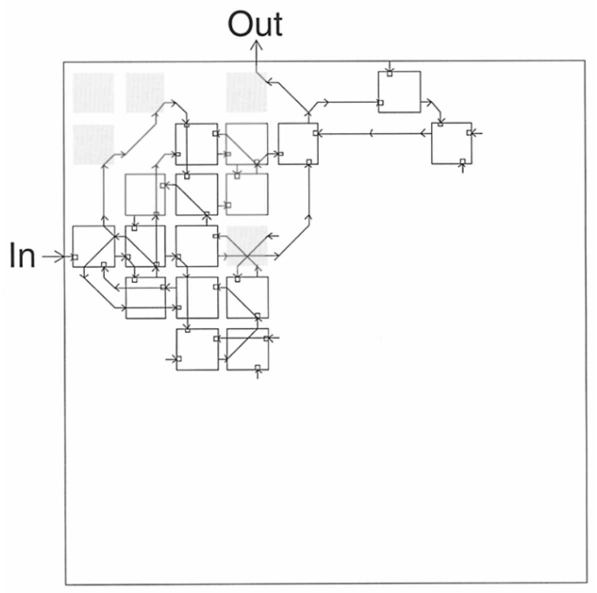


Fig. 19. The functional part of the circuit. Cells not drawn here can be clamped to constant values without affecting the circuit's behavior.

encouragement of small solutions. This may be chance, or it may be a natural way to solve the problem. It may be that the mutation rate was sufficiently high that any larger functional circuit could not be maintained by selection (the error threshold [59]). Finally, the implicit search biases mentioned in Section I-A might have been at play.

The circuit's behavior is not brittle. Fig. 20 shows the average output voltage (measured using the analogue integrator over a period of 5 s) for input frequencies from 31.25 kHz to 0.625 kHz. For input frequencies $\geq$4.5 kHz the output stays at a steady +5 V, and for frequencies $\leq$1.6 kHz at a steady 0 V. Thus, the test frequencies (marked F1 and F2 in the figure) are correctly discriminated with a considerable margin for error. As the frequency is reduced from 4.5 kHz, the output begins to rapidly pulse low for a small fraction of the time; as the frequency is reduced further the output spends more time at 0 V and less time at +5 V, until finally resting at a steady 0 V as the frequency reaches 1.6 kHz. Beyond this range, the output stays at steady 0 V for inputs down to 0 Hz (dc) and at steady +5 V for inputs up to several MHz. These properties might be called generalization and extrapolation, but are fortuitous: no steps were taken to encourage them. It may be that this is a natural response for a dynamical system of this class, but not enough data is yet available to be sure.

Fig. 20 also shows the circuit's behavior at temperatures outside the range experienced during the evolutionary experiment. The high temperature was achieved by placing a 60 W light bulb near the chip, the low temperature by opening all of the laboratory windows on a cool breezy evening. Varying the temperature moves the frequency response curve to the left or right, so once the margin for error is exhausted the circuit no longer behaves perfectly to discriminate between F1 and F2.

In the examples given here, at 43.0 °C the output is not steady at +5 V for F1, but is pulsing to 0 V for a small fraction of the time. Conversely, at 23.5 °C the output is not a steady 0 V for F2, but is pulsing to +5 V for a small fraction of the time.

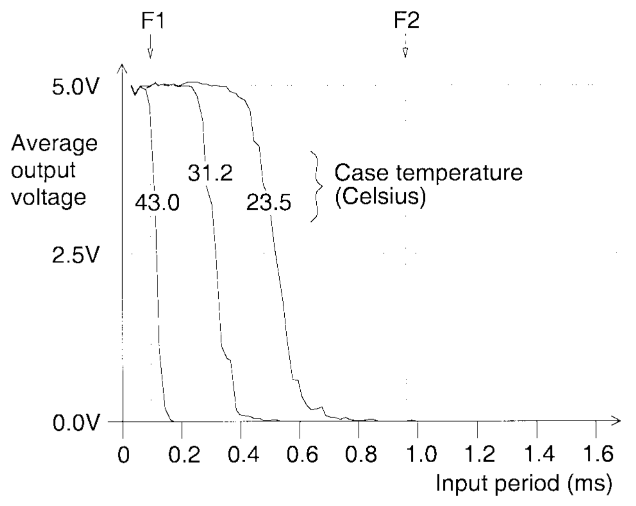


Fig. 20. The frequency and thermal response of the final circuit. F1 and F2 are the two frequencies that the circuit was evolved to discriminate. For ease of implementation, their exact periods were actually 0.096 ms (10.416 kHz) and 0.960 ms (1.042 kHz), respectively.

However, the circuit operates perfectly over the ±5 °C range of temperatures to which the population was exposed during evolution, despite its only available time reference being the natural dynamical behaviors of the components, which are temperature dependent.

C. Analysis

A useful circuit behavior has been evolved. The circuit has some attractive engineering properties, chiefly its small size. Its externally observable behavior has been characterized with respect to different inputs and operating temperatures, and found to be satisfactory within particular ranges (albeit a small temperature range and only using this particular FPGA device; see Section V-A). It might be thought that this knowledge would be enough to allow the circuit to be employed confidently in an application respecting these observed limitations, but this is not so.

To advance research, or to learn new design ideas from the circuit, we need to be able to discern some of its principles of operation. Even if the goal is merely to evolve a circuit that works (and we do not need to know how), however, some degree of analysis may still increase its utility as an engineering product. In particular, if bounds on possible long-term changes in the circuit's behavior can be derived, then the circuit can be more widely applied with confidence. The need for extended consistent performance is difficult to accommodate within an evolutionary framework, because usually the tests for fitness measurement of candidate solutions (the bottleneck in the evolutionary process) are as brief as possible. The evolutionary approach can be made more viable if, through analysis, it can be predicted that a circuit will perform adequately in the long term, even though it was never tested for long during its evolution.

There are two components to long-term stability of behavior. First, the circuit must be insensitive to certain variations in its implementation or environment: robust with respect to a necessary operational envelope. There can be a temporal aspect to robustness as it includes thermal drift over time, noise, aging effects in semiconductor devices and integrated circuits, and so on. Second, it is possible for even simple dynamical systems to display intermittent behavior over long time scales [60]. This is not due to any external fluctuations, but is a property of the system's own dynamics. Circuits can be constructed that will inevitably—though after a long period of normal operation—suddenly and unpredictably change in their qualitative mode of behavior, possibly forever, or perhaps to return to normal operation for another long interval [61]. An evolutionary algorithm, unless using debilitatingly long fitness evaluation tests, would be blind to this pathological behavior and could present such a circuit as a solution to the engineering specification. Inherently erratic dynamics of this kind can also interact with the temporal aspects of the operational envelope. If analysis can provide reassurance against long-term sporadic misbehavior, the circuit is rendered more useful.

In critical applications, complex circuits can be embedded within an error-recovering framework [62]. The error recovery mechanisms themselves can be simpler and perhaps verified formally. For example, if a failure condition is detected, the circuit responsible could be automatically reset to a safe initial state. The more a circuit's potential failure modes are understood, the more feasible it becomes to construct a resilient system containing it.

Analysis of exotic evolved circuits is different to that undertaken as part of orthodox design. At an abstract level, the appropriate tools are sometimes more akin to neuroscience than to electronic engineering. It is especially important to recognize that an evolved system may not have a clear functional decomposition. A functional analysis decomposes the system into semi-independent subsystems with separate roles; the subsystems interact largely through their functions and independently of the details of the mechanisms that accomplish those functions [63]. Systems designed by humans can usually be understood in this way, because of the "divide and conquer" approach universally adopted to tackle complex designs.

Although an evolved system may have particular functions localized in identifiable subsystems, this is not always so. Dynamic systems theory [64] provides a mathematical framework in which systems can be characterized without a functional decomposition. Hence, what to many people is the essence of understanding—being able to point at parts of the whole and say what function they perform—is not always possible for evolved systems. In this case, more precisely formulated questions regarding the organization of behavior must replace fuzzy notions of understanding or explanation rooted in functional decomposition. In our case, these questions are centered around the suitability of an evolved circuit for engineering applications. Addressing these questions, such as those regarding long-term dynamics, is what we mean by analysis.

The successful action of a circuit can be considered as a property of the interface between its inner mechanisms and the external environment [63]: the inner has been adapted so that the behavior at the interface satisfies the specification.

Observations at the interface (e.g., at input and output connections) during normal circuit operation may reveal little about the inner mechanisms, but instead will largely reflect the demands of the specification. Analysis therefore requires internal probing, and/or observation under abnormal conditions, either internal or external.

There are surprisingly many tactics that can be used to piece together an analysis.

- *Probing and abnormal conditions*. Abnormal conditions include: manipulation of the input signals, varying temperature or power-supply voltage, replacing parts of the circuit with alternative or nonfunctional pieces, and injecting externally generated signals at internal points. Monitoring an internal voltage always has some side effect, often placing a mostly reactive load at the probing point. This may have negligible consequences, but potentially perturbs the measurement or even stops the circuit from working altogether. Probing internal signals of a circuit implemented on an FPGA can require routing extra connections to reach the external pins of the device, with a danger of further disrupting the circuit under study. Advanced noncontact measurement equipment such as the electric-potential microscope [65] should prove powerful for scientific understanding, but is inconvenient for regular engineering projects, requiring the surface of the silicon die to be exposed.
- *Mathematical techniques*, including standard electronics theory, are preferable for their rigor and generality. If a whole unconventional evolved circuit is mathematically intractable, there may still be parts of it which yield.
- *Simulation* of a circuit allows rapid and extensive interactive exploration. Circuits evolved not in simulation, but using real reconfigurable hardware, may rely on detailed hardware properties not easily modeled in a simulation. Attempts at simulation can at least help to clarify the extent of this dependence.
- *Synthesis:* a circuit can be implemented using alternative electronic devices. For example, a circuit evolved on a single very large scale integration (VLSI) reconfigurable chip might then be constructed out of a number of hard wired small-scale chips. This provides easy access for probing and manipulation of internal signals and again can clarify what aspects of the hardware are important to the circuit's operation.
- *Power consumption* for the most common VLSI technology (CMOS) is related to the rate of change of the internal voltages. After removing any power-supply smoothing capacitors, power consumption can be monitored with high temporal precision, while the circuit is exposed to test conditions.
- *Electromagnetic emissions*, resulting from rapidly changing electrical signals, can sometimes be detected using a tuned radio receiver. Circuit activity within a chip, which might be difficult to monitor directly, can thus be roughly characterized.
- *Evolutionary history:* The mechanism underlying a task-achieving behavior may be more apparent soon after its evolutionary origin, rather than after evolution has refined it closely to match the specification. It may be possible to identify the innovation (perhaps caused by one or more mutations) giving rise to the behavior's origin in an ancestor, and to relate this to the operation of the final circuit.
- *Population diversity:* Sometimes there can be several slightly different (but related) forms of high-fitness circuits in an evolutionary population, which can help to reveal the basic mechanisms used.

Although unconventional evolved circuits can seem dauntingly unfamiliar, the analyst is far from powerless. We now apply these tactics to the evolved tone-discriminator, which is probably the most bizarre, mysterious, and unconventional unconstrained evolved circuit yet reported. The aim is to explore how analysis may be able to abate some of the worries associated with employing very unusual evolved circuits in an engineering application.

From the observations made so far, one could only speculate as to the circuit's means of operation, so unusual are its structure and dynamics. It was clear that continuous time played an important role. If the circuit was configured onto a different, but nominally identical, FPGA chip (not used during evolution), then its performance was degraded. If the temperature is raised or lowered, then the circuit still works, but the frequencies between which it discriminates are raised or lowered. (Digital circuits usually display unchanged behavior followed by brittle failure on approaching their thermal limits [66].) These initial observations warranted a concerted application of our tactics for unconventional analysis.

Recall that at intermediate frequencies, the circuit's output alternates rapidly between low and high voltages (Fig. 20), otherwise it is steady high or low. This binary behavior of the output voltage suggested that perhaps part of the system could be understood in digital terms. By temporarily making the assumption that all of the FPGA cells were acting as Boolean logic gates in the normal way, the FPGA configuration could be drawn as the logic circuit diagram of Fig. 21.

The logic circuit diagram shows several continuous-time recurrent loops (breaking the digital design rules), so the system's behavior is unlikely to be fully captured by this Boolean level of abstraction. It contains, however, many "canalyzing" functions [67], such as AND and OR functions, where one input can assume a value that makes the other inputs irrelevant. It so happens that whenever our circuit's input is **1**, all of the recurrent loops in parts A and B are broken by a cascade of canalyzing functions. Within 20 ns of the input becoming **1**, A and B satisfy the digital design rules, and all of their gates deterministically switch to fixed, static logic values, staying constant until the input next goes to **0**.

Part C of the circuit is based around a 2:1 multiplexer. When Part B is in the static state, the multiplexer selects the input marked "1" to be its output. This input comes from the multiplexer's own output via an even number of inversions, resulting in no net logic inversion but a time delay of around 9 ns. Under certain conditions, it is possible for such a loop to oscillate (at least transiently), but the most stable condition is for it to settle down to a constant logic state. The output of the whole circuit is taken from this loop. As this part C loop

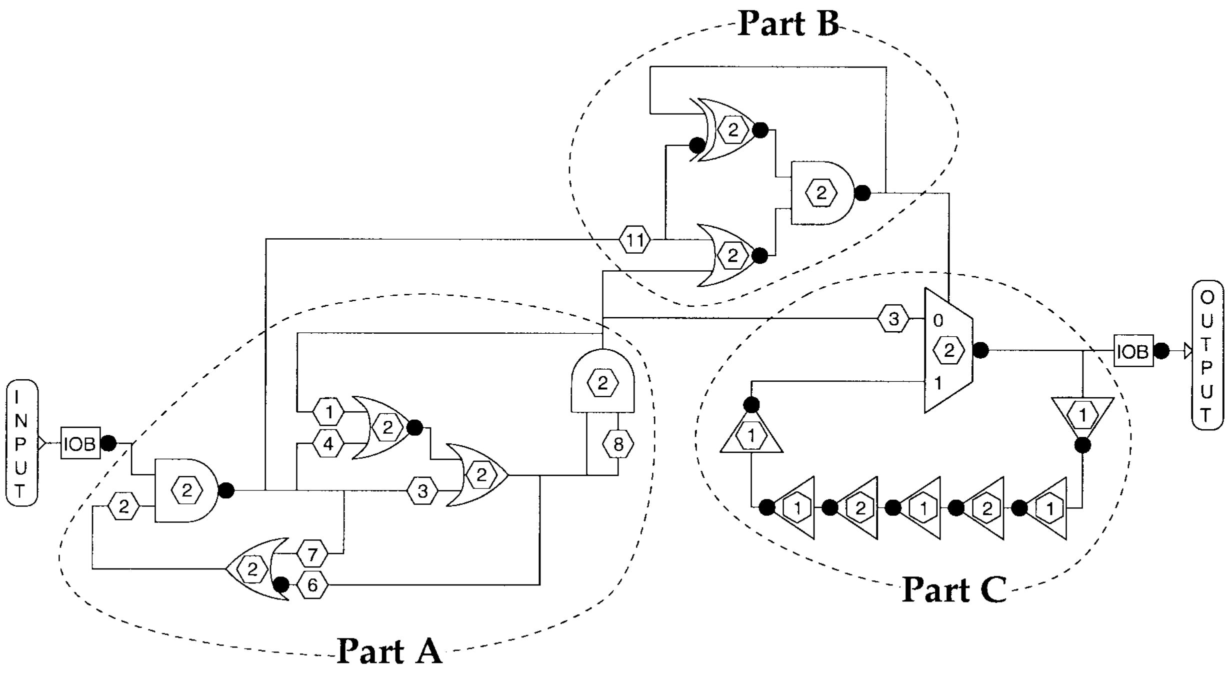


Fig. 21. The logic circuit representation. The numbers in hexagons are rough estimates of time delays (in ns), based on the maximum values specified by the manufacturer. IOB refers to the inverting input/output blocks that interface the edges of the reconfigurable array to the physical pins of the chip.

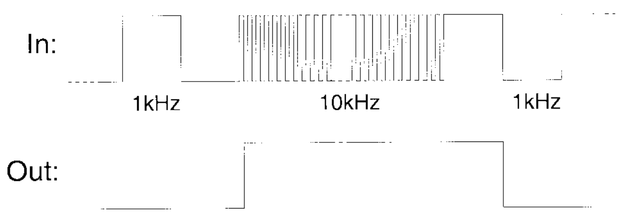


Fig. 22. The timing of the evolved circuit's response to a change in input frequency.

provides the only possibility for circuit activity during a high input, the next step in the analysis was to inspect the output very carefully while applying test inputs.[5]

We had already observed that the output only ever changes state (high $\Rightarrow$ low or low $\Rightarrow$ high) on the falling edge of the input waveform (Fig. 22). It was then discovered that the output also responds correctly to the width of single high-going pulses. Fig. 23 shows a low $\Rightarrow$ high output transition occurring after a short pulse; further short pulses leave the output high, but a single long pulse will switch it back to the low state. The output assumes the appropriate level within 200 ns after the falling edge of the input. The circuit does not respond to the width of low-going pulses, and recognizes a high-going pulse delimited by as little as 200 ns of low input at each end of the pulse. The output is perfectly steady at logic **1** or **0**, except for a brief oscillation during the 200 ns "decision time" which either dies away or results in a state change.

This is astonishing. During the single high-going pulse, we know that parts A and B of the circuit are reset to a static state within the first 20 ns (the pulse widths are vastly longer: 500 μs and 50 μs correspond to 1 kHz and 10 kHz). Our observations at the output show that Part C is also in a static state during the pulse. Yet somehow, within 200 ns of the end of the pulse, the circuit "knows" how long it was, despite being completely inactive during it.

This is hard to believe, so we have reinforced this finding through many separate types of observation, and all agree that the circuit is inactive during the pulse. Power consumption returns to quiescent levels during the pulse. Many of the internal signals were (one at a time) routed to an external pin and monitored. Sometimes this probing altered (or destroyed) the circuit's behavior, but we have observed at least one signal from each recurrent loop while the circuit was successfully discriminating pulse-widths, and there was never activity during the pulse. We were concerned that perhaps, because of the way the gates are implemented on the FPGA, it was possible that glitches (very short-duration pulses) were able to circulate in the circuit while our logic-analysis predicts it should be static; possibly these glitches could be so short as to be unobservable when routed to an external pin. Hence, we hand-designed a high-speed glitch-catching circuit (basically a flip-flop) as a configuration of two FPGA cells. Glitches sufficiently strong to circulate for tens of microseconds could be expected to trigger the glitch-catcher, but it detected no activity in any of the recurrent loops during an input pulse.

The circuit is not relying on influences from outside of the chip. Once the configuration had been downloaded to the device, it could be detached from all external circuitry. The only connections to the chip were then power-supply wires from a 6 V battery and shielded wires for the input and output, all directly wrapped onto the chip's pins (no socket, no decoupling capacitors). The whole isolated chip and battery

[5]The 20 MHz oscilloscope used for the earlier photographs of Fig. 15 was inadequate for analysis purposes. For the remainder, a Hewlett Packard 54542C four-channel digital storage oscilloscope was used, which samples at 2 Giga-samples/sec and is bandwidth limited to 500 MHz.

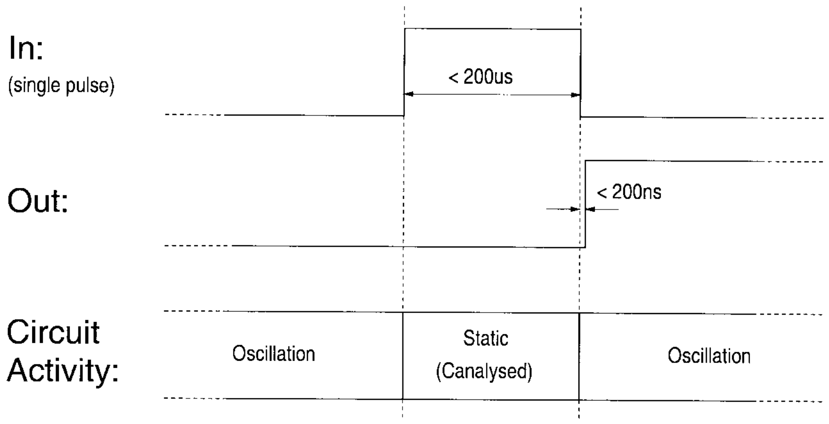


Fig. 23. The response to a single high-going pulse.

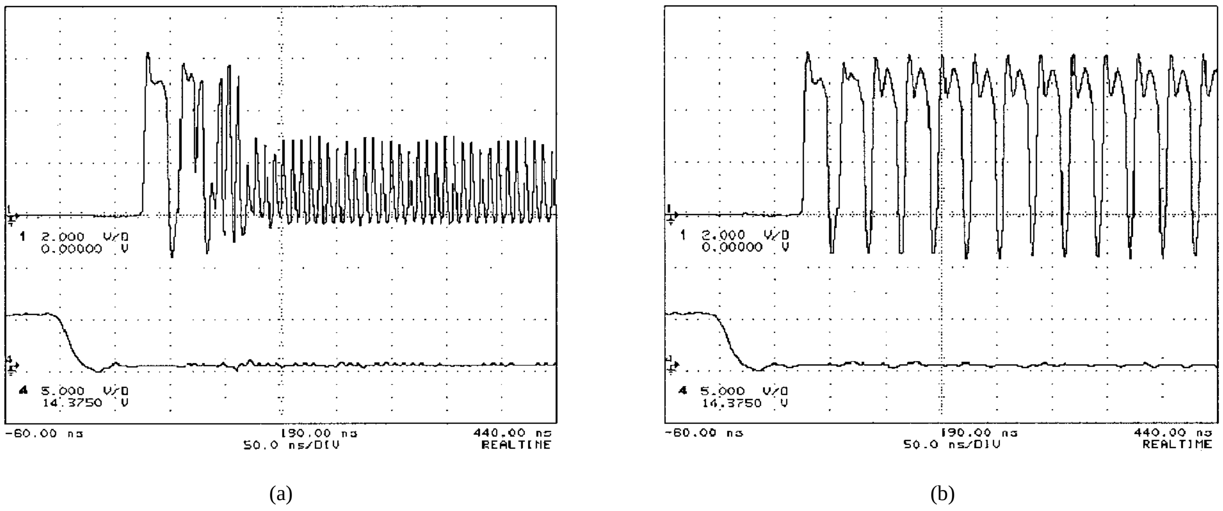


Fig. 24. Behavior of the first frequency-discriminating ancestor. The upper waveform is the output and the lower the input; we see the behavior immediately after the falling edge of a single input pulse. (a) Long pulse and (b) short pulse. The input to the FPGA is actually delayed ≈40 ns by an intervening buffer, relative to the wave seen here.

assembly could then be placed in a grounded metal box, and the circuit still displayed the correct behavior.

We performed a digital simulation of the circuit (using PSPICE), extensively exploring variations in time delays and parasitic capacitances. The simulated circuit never reproduced the FPGA configuration's successful behavior, but did corroborate that the transient as the circuit enters its static state at the beginning of an input pulse is just a few tens of nanoseconds, in agreement with our experimental measurements of internal FPGA signals, and according with the logic analysis. We then built the circuit out of separate CMOS multiplexer chips, mimicking the way that the gates are actually implemented by multiplexers on the FPGA, and also modeling the relative time delays. Again, this circuit did not work successfully, and—despite our best efforts—never produced any internal activity during an input pulse.

We then went back to find the first circuit during the evolutionary run that responded at all to input frequency. Its behavior, originally shown at generation 650 in Fig. 15 using an inadequate oscilloscope, is enlarged in Fig. 24. During a pulse, the output is steady low. After the pulse, the output oscillates at one of two different frequencies, depending on how long the preceding pulse was. These oscillations are stable and long lasting. The differences are minor between this circuit and its immediate evolutionary predecessor (which displays no discrimination, always oscillating at the lower of the two frequencies). In fact, there are no differences at all in the logic circuit diagrams; the changes do things like alter where a cell's function takes an unused input from. This lends further support that circulating glitches are not the key: there was no change to the implementation of the recurrent loops.

We see bistable oscillations similar to Fig. 24 at internal nodes of part A of the final circuit. On being released from the canalyzed stable state, the difference in the first 100 ns of oscillatory behavior in part A is used by parts B and C to derive a steady output according to the pulse width. There is some initial state of the part A dynamics which is determined by the input pulse length. This initial state does not arise from any circuit activity in the normal sense: the circuit over the entire array of cells was stable and static during the pulse. It is a particular property of the FPGA implementation, as it is not reproduced in simulation or when the circuit is built from separate small chips. One guess is that the change in initial state results from some slow charge/discharge of an unknown parasitic capacitance during the pulse, but we cannot yet be sure.

We understand well how parts B and C use A's initial oscillatory dynamics to derive an orderly output, and have successfully modeled this in simulation. The time delays on the connections from A to B and C are crucial. This explains the influence of the grey cells, which are all immediately adjacent to (or part of) the path of these signals. Varying the time delays in the simulation produces a similar result to interfering with the grey cells. Mostly, the loop of part C serves to maintain a constant and steady output even while the rest of the circuit oscillates, but immediately after an input pulse it has subtle transient dynamics interacting with those of A and B.

D. Conclusion to Case Study 2

The results described here represent the state of the art in the exploration of radically new territories of design space. The circuit is small, but definitely not trivial. For a human designer to solve this problem using only 32 cells, with no clock or external components, would be very difficult indeed (if feasible at all).

The circuit vividly demonstrates the power of unconstrained evolution. With a freedom to explore rich structures and dynamics, evolution has been able to exploit the natural behaviors arising from the physics of the device. Analysis of such an evolved circuit enhances its utility, but requires novel approaches. There are numerous tactics that can be used to piece together answers to analysis questions even for a seemingly impenetrable circuit. We still do not understand fully how it works: the core of the timing mechanism is a subtle property of the VLSI medium. We have ruled out most possibilities: circuit activity (including glitch-transients and beat-frequencies), metastability [41], and thermal time constants from self-heating. Whatever this small effect, we understand that it is amplified by alterations in bistable and transient dynamics of oscillatory loops and in detail how this is used to derive an orderly near-optimal output. Certain peripheral cells fine-tune particularly sensitive time delays. On the key question of long-term consistency of behavior, we know that the entire FPGA circuitry is strongly reset to a deterministic stable logic state for every high half-cycle of the input waveform. Long-term pathologies are therefore highly unlikely, demonstrating that analysis effort can sometimes remove worries related to the use of highly unconventional circuits in practical applications.

It now seems indisputable that hypotheses H1 and H2 are true: "Evolutionary algorithms can explore some of the regions in design space that are beyond the scope of conventional methods." The fascinatingly alien tone-discriminator circuit was produced using a very basic evolutionary method, with no great difficulty other than to leave behind preconceptions of how electronics should be. The circuit gives a tantalizing glimpse of the theoretically possible engineering attractions, such as small size by finding forms and processes that are natural to the VLSI medium. However, it is invalid to make a direct comparison with conventionally designed circuits: these have a much larger operational envelope.

V. Towards Robust Unconventional Evolved Circuits

If the 10×10 region of the evolved tone-discriminator was translated to another region of the 64×64 array, on the same chip, then performance was degraded: sometimes slightly, sometimes completely. Similar failings are found if the evolved configuration is used on a different nominally identical FPGA, though in both cases if evolution was allowed to continue, the population quickly adapted to the characteristics of the new silicon. Combined with the relatively narrow range of operating temperatures, this unportability restricts the circuit to very esoteric applications. To claim that an unconventional circuit is better (hypothesis H3), in a practical sense, it must have a more comprehensive operational envelope. We now present three research tools currently in use to bring this about, or at least to illuminate the potentials and limitations of the unconstrained approach.

A. The Evolvatron

The "Evolvatron" is a tool that can provide an evolutionary selection pressure for robustness, without having to prejudge how this should be achieved. For example, [66] observes that analogs of many of the strategies for thermal robustness found in animals, from a behavioral level down to the molecular, could also arise in unconstrained evolved electronics subject to this selection pressure. Some of these strategies are very different to those practiced in conventional electronics design.

Section II described how orthodox methods bring about robustness by adopting general design constraints. In contrast, the evolutionary goal is to produce mechanisms for robustness that are tailored to the task, the structure and dynamics of the circuit, and—in turn—the natural physical resources of VLSI. This approach is less restrictive to design space exploration, and can potentially find holistic strategies for robustness that are superior in their particular domain. By holistic we mean a strategy not only tailored to the situation, but which also emerges at the system-level, not necessarily demanding robustness of all of the parts. Here, robustness is part of the target of the evolutionary design process, whereas for conventional methods it is mostly a constraint placed upon the design process itself.

The Evolvatron is shown in Fig. 25 and is fully described in [68]. It consists of an expandable collection of XC6216 FPGA's, which are maintained in different conditions. Fig. 26 summarizes the conditions currently provided in the pictured machine. Using five chips, from two different foundries, a selection is provided of thermal conditions, power-supply voltages, output loads, host $\Rightarrow$ FPGA interfaces, chip packages, and positionings of the 10×10 evolved region within the array.

For a fitness evaluation, a circuit is downloaded to each of the chips, which are then tested simultaneously at the target task. The evaluation function combines the five scores so as to give a measure of the circuit's ability to perform the task in all of these conditions: the selection pressure for

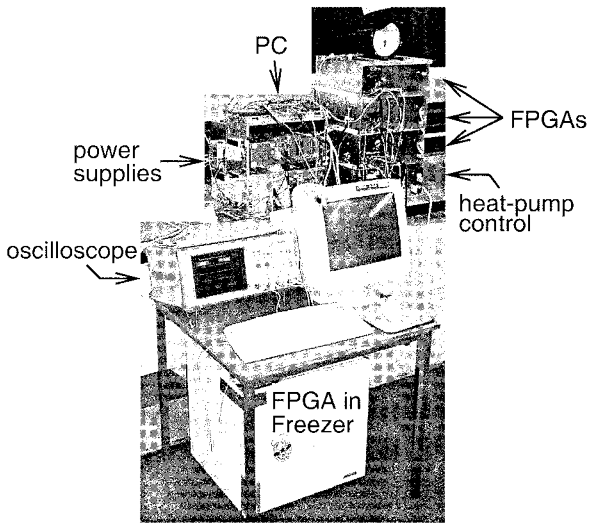


Fig. 25. The Evolvatron.

robustness. Preliminary results for the tone discrimination task are encouraging [68]. The primary research questions are the following.

- What resources are needed for robustness? Are stable off-chip components required? Is it necessary to provide a stable oscillation as an extra input to the evolving circuits? Such a clock would be a resource to be used in any way (or ignored) as appropriate, rather than an enforced constraint on the system's dynamics as in synchronous design.
- Can generalization over an entire, practical, operational envelope be achieved through evolution in a relatively small number of different conditions?
- Given the experimental arrangement described herein, is the evolution of a robust circuit more difficult than of a fragile circuit evolved on just one FPGA? (An apparently harder task is not necessarily more difficult for evolution, depending on the pathways available for evolutionary change.) Does it make sense gradually to increase the diversity and span of the operational envelope during evolution, or should the population be exposed to a representation of the complete operational envelope right from the beginning?
- If robust circuits *are* evolved, how do they work? Do they look more like conventional circuits? Are they still smaller?

B. The Evolvable Motherboard

We now introduce a research tool designed to allow exploration of design space further than is possible with commercial FPGA's. The tool [henceforth referred to as the evolvable motherboard (EM)] allows a large variety of electronic components to be used as the basic active elements and has an interconnection architecture such that any component pin can be independently connected to any other. Interconnections are directly accessible to test equipment, facilitating analysis of circuits configured on the EM. Fig. 27 is a simplified diagram of one corner of the EM and the plug-in daughterboards containing the basic elements. The diagonal lines represent digitally controlled analogue switches which allow row/column interconnection. The minimum number s of switches required to ensure all possible combinations of interconnections between basic elements is equal to the number of different pairs of the total of their pins

$$s = \frac{n(n-1)}{2} \qquad (3)$$

where n is equal to the total number of basic element pins.

Equation (3) can be realized using a triangular matrix of n rows $\times$ n columns, approximated on the EM using commercial analogue crosspoint switch arrays. Each daughterboard takes up to eight lines on the switch matrix, plus a further eight connections to allow for power lines and I/O, which may be required by components such as operational amplifiers or digital potentiometers. EM's have been constructed using $n = 48$ (Fig. 28), admitting up to six daughterboards. Expansion ports are provided so that several EM's can be daisy-chained together.

Connections made using the analogue switches have resistance and capacitance, hence forming an integral part of any circuit configured. In total, approximately 1500 switches are used, giving a search space of 10^{420} possible circuits. The on resistance of the analogue switches prevents configurations that short the power rails from damaging the EM provided the power supply is less than 3 Vdc. Using an ISA interface (not shown), the switches can be programmed by direct writes to a PC's internal I/O ports, allowing circuits to be instantiated in hardware in a very short time (<1 ms).

The EM was conceived to help provide insights into choosing the basic element type and interconnection architecture of an FPGA ideally suited to circuit design using artificial evolution and to aid analysis of bizarre evolved circuits whose operation could not be explained by function-level models. Research is currently in progress using transistors, multiplexers, and operational amplifiers as basic elements, but results presented in this paper are restricted to the use of bipolar transistors. By catering for all possible interconnections, a variety of more restrictive architectures can be evaluated for a given EA by the appropriate choice of genotype-phenotype mapping. While simple circuits have been successfully evolved using the full complement of switches (by directly mapping each genotype bit to a different switch), this is not generally appropriate since candidate solutions tend to short out the basic elements [69]. The following example illustrates the use of an interconnection architecture chosen to reflect the connectivity found in conventional circuits.

The task was to evolve a circuit to minimize the ac error between the output and amplified input voltages, using the fitness measure

$$f = -\frac{1}{500}\sum_{i=1}^{500}\left|a\left(V_{\mathrm{in}_i} - O_{\mathrm{in}}\right) - \left(V_{\mathrm{out}_i} - O_{\mathrm{out}}\right)\right| \qquad (4)$$

where a is the desired amplification factor, V_{in_i} and V_{out_i} are the ith input and output voltage measurements, respectively, and O_{in} and O_{out} are the dc offsets of input and output, respectively. Amplification a was set to -10. The fitness measure equates to a simple inverting amplifier; however, it

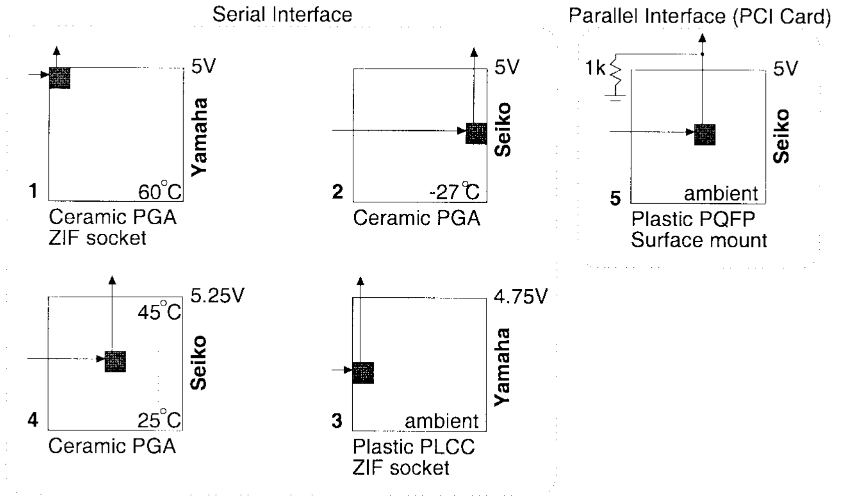


Fig. 26. An example of an achievable set of test points within an operational envelope.

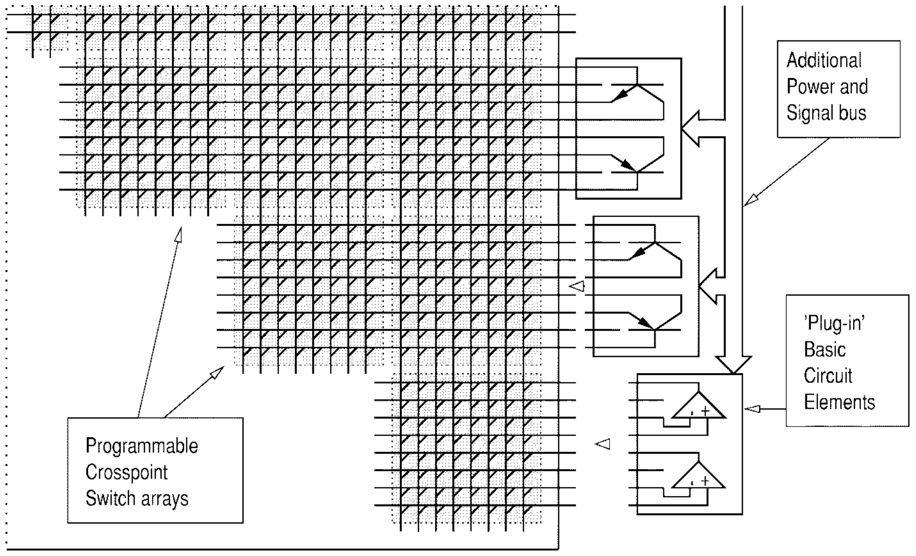

Additional Power and Signal bus
'Plug-in' Basic Circuit Elements
Programmable Crosspoint Switch arrays

Fig. 27. Simplified schematic of part of the evolvable motherboard.

is not intended to be a practical amplifier since the fitness measure makes no provision for phase shift, and only a single frequency was applied at the input during evaluation: a 1 kHz sine-wave of 2 mV peak-to-peak amplitude, offset at +1.4 Vdc. A rank based, generational genetic algorithm with elitism was used for all the runs, with population size 50. Genetic operators were mutation and single-point crossover, with mutation probability set at 0.01 per bit. The genotype is mapped to the motherboard switches so as to limit the quantity of switches on per row, so that the pins of active components are not too highly interconnected. This is consistent with many conventional circuits where each component pin is only connected to two or three other pins. In the encoding, each column is assigned a corresponding row. The genotype represents the switches a row at a time. For each row, one bit specifies whether the corresponding column is connected, followed by column address and connection bits for up to n additional switches. n was set to 3, and 48 rows/columns were used giving a genotype length of 1056 bits. The task was made nontrivial by denying evolution the use of components that would be considered essential for conventional design, in this case resistors and capacitors. This constraint is potentially useful for VLSI.

Fig. 29 is a circuit diagram typical of those obtained for the task during 20 runs of 8000 generations each. The circuits cannot be analyzed in the traditional manner, since the current gain of bipolar transistors (β) varies widely for different specimens of a given type. Conventional circuits are designed to rely only on this property being above some minimum value [27], whereas unconstrained evolution will exploit the actual value for this and other properties. It is therefore difficult to be certain from the diagram alone which transistors have an active role, and which are "junk." Using the EM, analysis is far simpler: unplugging each transistor

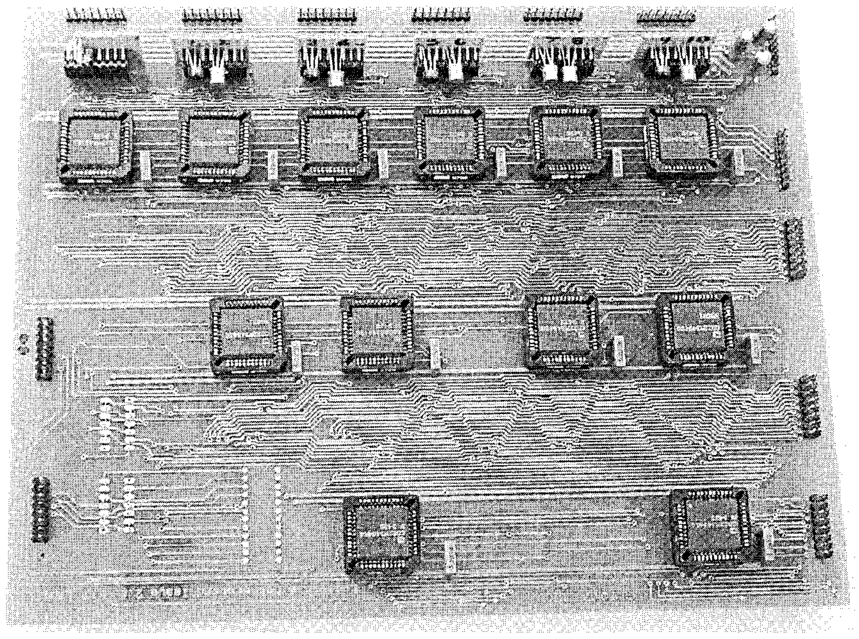


Fig. 28. An evolvable motherboard, with daughterboards of two transistors each attached.

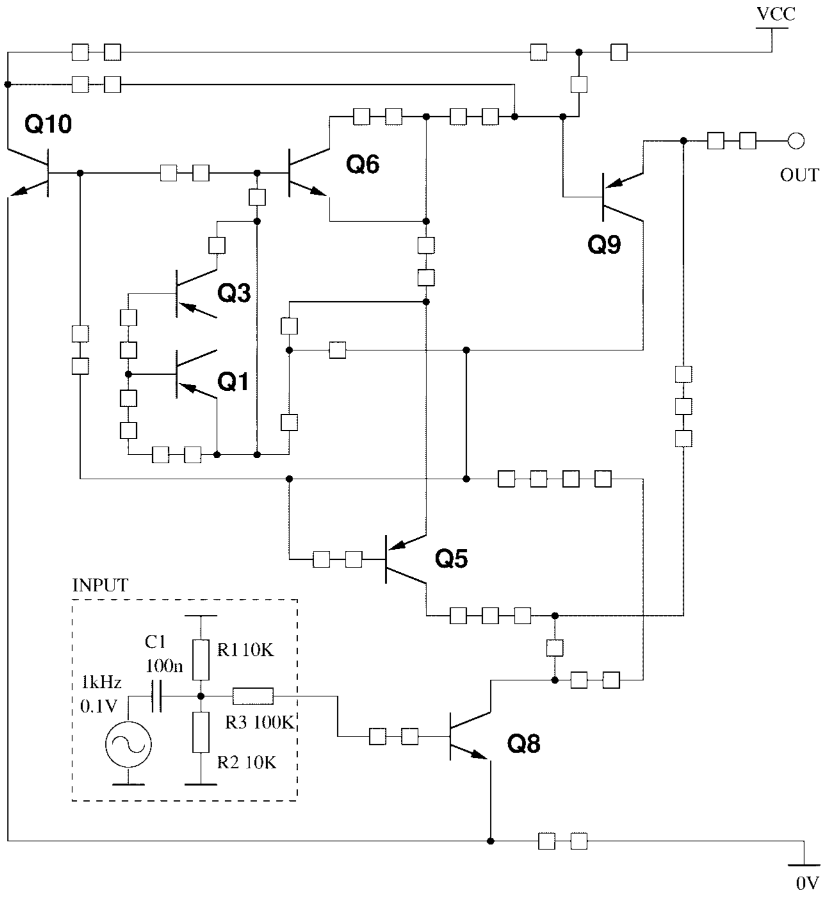


Fig. 29. A typical evolved amplifier. The small squares represent analogue switches turned on.

and reevaluating shows that only Q8 and Q10 are essential to the circuit's operation (Fig. 30). Measuring the voltage directly at the transistors' terminals reveals both are operating as emitter-followers. This simple example demonstrates the EM's potential for evolving and analyzing small circuits with arbitrary architectures and active elements, which are elaborate enough to be used as building blocks in analogue design. Currently, the EM's flexibility and observability is being

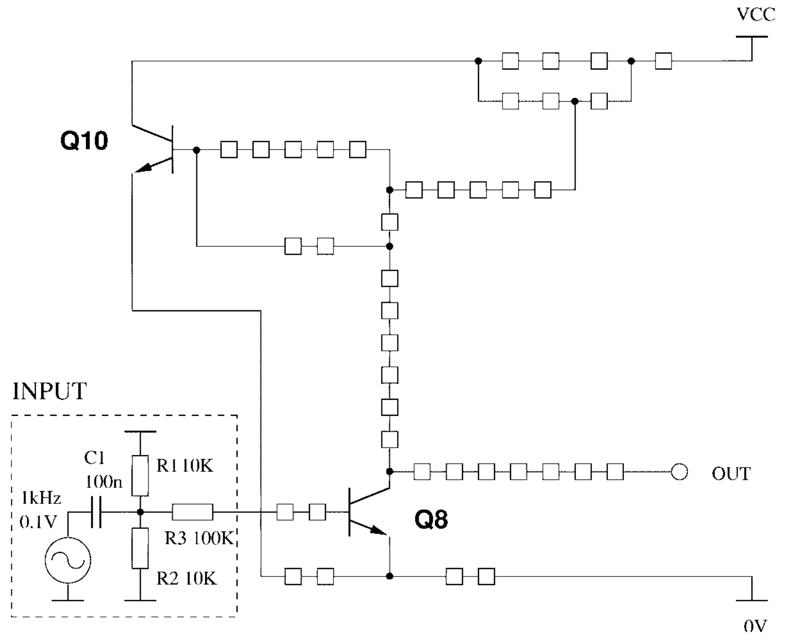


Fig. 30. A pruned circuit diagram of the amplifier.

used to study the topologies, dynamics, and failure modes, of unconventional evolved circuits.

C. Evolution in Simulation of Buildable Circuits

The use of real reconfigurable hardware ensures that the properties of the physical electronic medium are available without restriction, to be exploited by the evolved circuits. Evolution using a detailed simulation is also possible, but a simulator inevitably neglects some details, while spuriously affecting others, for all but the smallest of circuits. Simulations are attractive, being more controllable, observable, and in some cases faster (in others infeasibly slower), than using real reconfigurable hardware running in real time. Noise levels can be controlled in a simulation, which can affect the evolutionary dynamics [70]. In the following, we give an example of the special precautions that must be taken to ensure that bipolar transistor-level simulation results are valid: that the evolved circuits will actually work when built. The commercial simulator SMASH will be used, but the comments apply equally to the use of other well-known packages such as SPICE.

The most elementary precaution is only to allow components in the simulation that are really available. This not only implies the use of preferred values for resistors, capacitors, etc. [71], but also adjusting the transistor model parameters, such as the saturation current I_s and β, to match closely the components to be used in the real implementation.

It is also necessary to check during the simulations that the transistors would not be damaged by overcurrent conditions. This can be done by checking that the base-emitter voltages and collector currents do not exceed limits (usually ≈0.7 V, and around 100 mA for low-power transistors). Without this check, it is common to evolve a circuit that seems to work in the simulation, but which will instantly destroy the transistors at power-up if built.

As an example, consider the evolution using a GA of a transistor amplifier. The genotype was a linear string using integer encoding: it consisted of a separate gene corresponding to each component. The gene determines the nature, value and connecting points of the related component. Experimental details can be found in [10]. In this particular set of experiments, the genotype was made up of eight genes, with a total of ten connecting points available for the evolutionary algorithm to arrange the components. Half of these points are external ones, being connected to: a positive power supply (12 V), a negative power supply (−12 V), ground, the input signal, and the circuit output. The other five points are internal to the circuit. The biased input voltage represents a differential circuit input.

The fitness evaluation was based on a dc transfer analysis, in which the input signal was swept from the negative to the positive power supply voltages, in n increments of 100 mV. The function

$$\text{Fitness} = \text{Max}_{i=1}^{n-1} |V(i+1) - V(i)| \tag{5}$$

was used, where $V(i)$ describes the circuit output voltage as a function of the dc analysis index i, which spans the swept range of the input signal. This evaluation function aims to identify the maximum voltage gradients between consecutive input voltage steps; the higher the gradient, the larger the amplifier gain will be. Ideally, the amplifier transfer function should include two saturation regions separated by a narrow, linear and steep gain region [11].

In a first set of experiments, a penalty term was included in the fitness function to eliminate circuits presenting over-current or over-voltage conditions. In a second set, these restraints were not applied during the evolutionary process, but the final circuits were inspected before any attempt was made to build them. For the first set, there was a low success-rate for GA runs, but the solutions had a greater probability of working when built. For the second set, there was a higher apparent success rate for GA executions, but most of the evolved circuits would not work in practice. This finding gives a context to some reports in the literature of highly complex

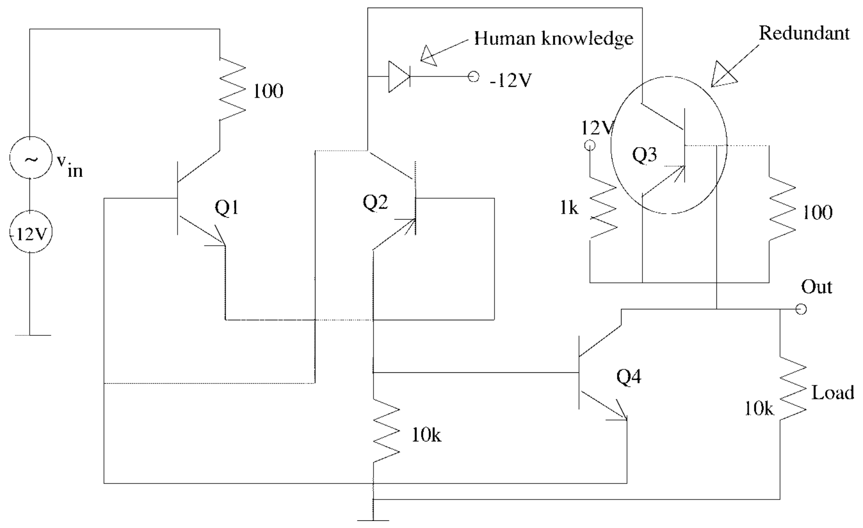


Fig. 31. The best evolved amplifier from the second set of GA runs.

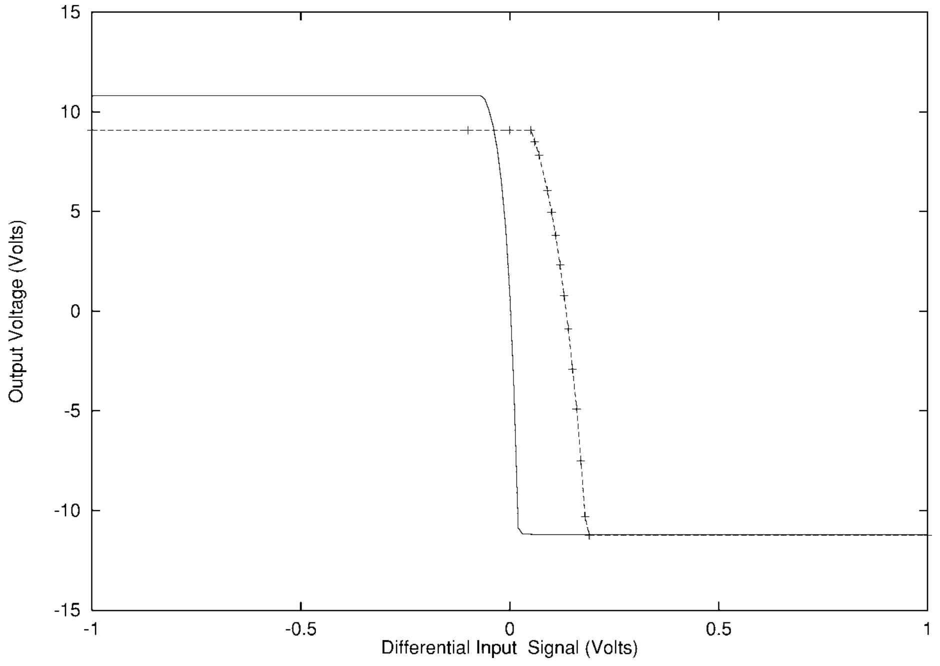


Fig. 32. Comparison between the dc transfer function obtained in simulation (solid line) and the one achieved by actually measuring the circuit output (dashed/crosses).

transistor circuits evolved in simulation not using preferred values, matched transistor models, or any check on over-current or over-voltage, and for which no attempt was made to implement using real components (e.g., [72]).

Fig. 31 depicts the schematic of the best amplifier synthesized in the second set of tests. In this run, approximately 10^5 individuals were evaluated, taking about 20 h running on one processor of a Sun Ultra Enterprise 2 workstation. Fig. 32 compares the dc transfer function obtained in simulation with that experimentally measured from the circuit actually implemented on a prototyping board (Fig. 33).

The topology of the circuit shown in Fig. 31 is very unconventional: the input is being applied to the collector, and not to the base, of transistor Q1. The circuit is not exactly equivalent to any standard transistor stage. Transistor Q3 is redundant.

By running an operating point analysis, it was discerned that transistors Q2 and Q4 were working as current amplifiers, whereas transistor Q1 was biased in its reverse region. Q1 and Q2 act so as to set the dc operating point of Q4, which is delivering the amplification of the overall circuit.

The diode shown was inserted after evolution to remove a bias voltage of -0.7 V from the circuit. The resistor values produced by the GA have also been slightly changed, using human knowledge, to improve the circuit linearity. Interactive involvement from an expert is not necessarily undesirable, but in cases such as this, linearity could alternatively be improved by a further application of an optimization algorithm to the component values within the evolved structure.

From Fig. 32, it can be observed that both the implemented and the simulated versions behave as inverting amplifiers. The

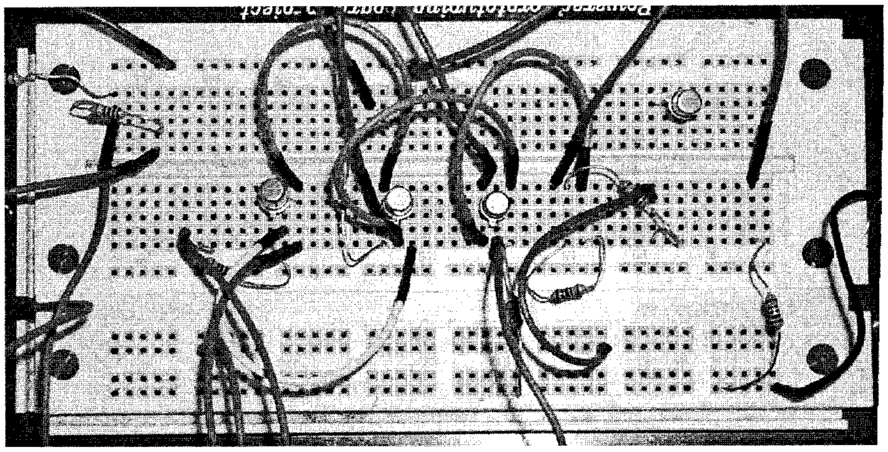


Fig. 33. The evolved circuit built on a prototyping breadboard for testing.

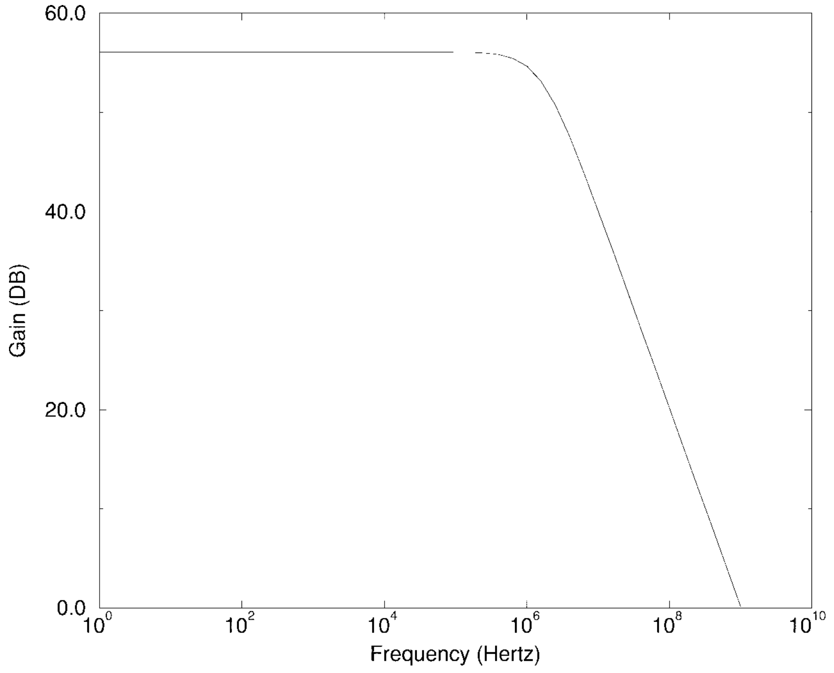


Fig. 34. Frequency response of the evolved amplifier (in simulation).

shift between the responses is due to a mismatch between the real diode parameters and the default ones employed by the simulator. An important feature of this amplifier, which had to drive a 10 kΩ resistive load during all tests, is that the saturation voltages approach the power supply values, an enhancement over our previous results [10].

At first glance, it might seem that this circuit is comparable to a commercial bipolar operational amplifier, such as the NE5534 [11]. The latter has a gain of 100 dB and a gain-bandwidth product of 10 MHz when driving a capacitive load of 100 pF, while the evolved circuit has a gain of 56 dB and a gain-bandwidth product of over 100 MHz for the same load, with both circuits dissipating around 100 mW. Fig. 34 depicts the frequency response of the evolved amplifier. Such a comparison is not warranted, however, because the specification of the NE5534 includes good performance by many other criteria. These criteria were not included in our experiment but are necessary for a useful op-amp [73]. A promising multicriteria evolutionary method is given in [15]; here we simply aimed to show that if simulations are to be used at all, then buildability must be a primary criterion.

VI. Conclusion

Three simple and precise hypotheses were addressed.

H1) That conventional methods can only work within constrained regions of electronics design space has been shown first by characterizing what conventional design practices actually are, and secondly by exhibiting an evolved circuit obviously beyond them: the tone discriminating circuit of Case Study 2.

H2) That evolutionary algorithms can explore some of the regions of design space that are beyond the scope of conventional methods has been shown explicitly. In Case Study 1, the DSM robot controller, evolution was seen to exploit the enhanced behavioral capabilities of a conventional architecture, once some constraints from digital design methods were relaxed. In Case Study 2, given the freedom, an evolutionary process produced a working circuit with a structure and dynamics foreign to orthodox design and analysis.

H3) That evolutionary algorithms can in practice produce designs beyond the scope of conventional methods, and that are better, has been signalled repeatedly but not demonstrated conclusively. On theoretical grounds, there is greater opportunity to find the forms and processes that naturally exploit the properties of the electronic medium, and this seems to happen in practice. Nonbehavioral requirements can be integrated into the design process more seamlessly, and an example was seen when the robot controller was evolved to be tolerant to SSA faults, without the explicit incorporation of redundant parts. There is the promise of the evolution of means for robustness that are tailored to the task, the circuit, and the medium. Hardware and software tools were presented to illuminate the way forward in scientific investigation. H3 is at least a good working hypothesis.

What initially seemed daring hypotheses are now either matter-of-fact, or within reach. From the vastness of design space, practically useful novel regions beckon.

Acknowledgment

The authors are indebted to P. Husbands, I. Harvey, and J. Gray and to the following for support: EPSRC, British Telecommunications, Xilinx, Hewlett-Packard, Zetex, and Motorola.

References

[1] A. Sloman, "The 'semantics' of evolution: Trajectories and trade-offs in design space and niche space," in *Progress in Artificial Intelligence: Proc. 6th Ibero-American Conf. AI IBERAMIA'98* (Lecture Notes in Computer Science, vol. 1484), H. Coelho, Ed. Berlin, Germany: Springer-Verlag, 1998, pp. 27–38.

[2] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 67–82, Apr. 1997.

[3] H.-P. Schwefel and G. Rudolph, "Contemporary evolution strategies," in *Advances in Artificial Life: Proc. 3rd Eur. Conf. Artificial Life* (Letcure Notes in Artifical Intelligence, vol. 929), F. Morán, A. Moreno, J. J. Merelo, and P. Chacon, Eds. Berlin, Germany: Springer-Verlag, 1995, pp. 893–907.

[4] M. Eigen, "New concepts for dealing with the evolution of nucleic acids," in *Proc. Cold Spring Harbor Symposia on Quantitative Biology*, Cold Spring Harbor Laboratory, Cold Spring Harbor, NY, 1987, vol. LII.

[5] M. A. Huynen and P. Hogeweg, "Pattern generation in molecular evolution: Exploitation of the variation in RNA landscapes," *J. Mol. Evol.*, vol. 39, pp. 71–79, 1994.

[6] A. Thompson, "Evolving inherently fault-tolerant systems," in *Proc. Instn Mechanical Eng.*, 1997, part I, vol. 211, pp. 365–371.

[7] I. Harvey, "Cognition is not computation: Evolution is not optimization," in *Proc. 7th Int. Conf. Artificial Neural Networks ICANN'97* (Lecture Notes in Computer Science, vol. 1327), W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, Eds. Belin, Germany: Springer-Verlag, 1997, pp. 685–690.

[8] G. M. Kunkel, "EMC system design: A systematic methodology," *Compliance Eng.*, vol. XIII, no. 5, July–Aug. 1996.

[9] M. L. Maher and J. Poon, "Modeling design exploration as co-evolution," *Microcomput. Civil Eng.*, vol. 11, pp. 192–207, 1996.

[10] R. S. Zebulum, M. Vellasco, and M. A. Pacheco, "Analog circuits evolution in extrinsic and intrinsic modes," in *Proc. 2nd Int. Conf. Evolvable Systems (ICES'98)* (Lecture Notes in Computer Science, vol. 1478), M. Sipper, D. Mange, and A. Pérez-Uribe, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 154–165.

[11] K. R. Laker and W. M. C. Sansen, *Design of Analog Integrated Circuits and Systems*. New York: McGraw-Hill, 1994.

[12] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evol. Comput.*, vol. 3, no. 1, pp. 1–16, 1995.

[13] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evol. Comput.*, vol. 4, no. 1, pp. 1–32, 1996.

[14] R. S. Zebulum, M. A. Pacheco, and M. Vellasco, "Artificial evolution of active filters: A case study," in *Proc. 1st NASA/DoD Workshop on Evolvable Hardware*. Los Alamitos, CA: IEEE Comput. Soc. Press, 1999, pp. 66–75.

[15] R. S. Zebulum, M. A. Pacheco, and M. Vellasco, "A multi-objective optimization methodology applied to the synthesis of low-power operational amplifiers," in *Proc. XIII Int. Conf. Microelectronics and Packaging*, vol. 1, I. J. Chueiri and C. A. dos R. Filho, Eds. Brazilian Microelectronics Society, 1998, pp. 264–271.

[16] T. Arslan, D. H. Horrocks, and E. Ozdemir, "Structural synthesis of cell-based VLSI circuits using a multi-objective genetic algorithm," *Electron. Lett.*, vol. 32, no. 7, pp. 651–652, Mar. 1996.

[17] R. Rogenmoser, H. Kaeslin, and T. Blickle, "Stochastic methods for transistor size optimization of CMOS VLSI circuits," in *Proc. 4th Int. Conf. Parallel Problem Solving from Nature (PPSN IV)* (Lecture Notes in Computer Science, vol. 1141), W. Ebeling, I. Rechenberg, H.-P. Schwefel, and H.-M. Voigt, Eds. Berlin, Germany: Springer-Verlag, 1996, pp. 849–858.

[18] R. S. Martin and J. P. Knight, "Power-profiler: Optimizing ASIC's power consumption at the behavioral level," in *Proc. 32nd Design Automation Conf.*, Assoc. Computing Machinery, 1995, pp. 42–47.

[19] A. M. Hill and S.-M. Kang, "Genetic algorithm based design optimization of CMOS VLSI circuits," in *Proc. 3rd Conf. Parallel Problem Solving from Nature (PPSN III)* (Lecture Notes in Computer Science, vol. 866), Y. Davidor, H.-P. Schwefel, and R. Männer, Eds. Berlin, Germany: Springer-Verlag, 1994, pp. 546–555.

[20] R. S. Martin and J. P. Knight, "Genetic algorithms for optimization of integrated circuits synthesis," in *Proc. 5th Int. Conf. Genetic Algorithms (ICGA'93)*, S. Forrest, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 432–438.

[21] G. Gopalakrishnan and V. Akella, "VLSI asynchronous systems: Specification and synthesis," *Microprocess. Microsyst.*, vol. 16, no. 10, pp. 517–526, 1992.

[22] K. K. Likharev and T. Claeson, "Single electronics," *Sci. Amer.*, vol. 266, no. 6, pp. 50–55, 1992.

[23] R. C. Johnson, "Full speed ahead—Mead sees a future without boundaries," *Elect. Eng. Times*, vol. 1028, Sept. 30, 1998.

[24] C. A. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison Wesley, 1989.

[25] C. Mead, "Time—The essential dimension," in *Proc. Int. Joint Conf. Neural Networks*. Hillsdale, NJ: Lawrence Erlbaum, 1990, vol. II, p. 25.

[26] R. Sarpeshkar, "Analog versus digital: Extrapolating from electronics to neurobiology," *Neural Comput,*, vol. 10, no. 7, pp. 1601–1638, 1998.

[27] P. Horowitz and W. Hill, *The Art of Electronics*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1989.

[28] R. A. Rutenbar, "Analog design automation: Where are we? Where are we going?," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1993, pp. 13.1.1–13.1.8.

[29] T. Blickle, *Theory of Evolutionary Algorithms and Application to System Synthesis*, TIK-Schriftenreihe Nr. 17, vdf, Hochsch.-Verl. an der ETH, Swiss Federal Institute of Technology, Zürich, 1997.

[30] M. Murakawa, S. Yoshizawa, I. Kajitani, T. Furuya, M. Iwata, and T. Higuchi, "Hardware evolution at function level," in *Proc. 4th Int. Conf. Parallel Problem Solving from Nature (PPSN IV)* (Lecture Notes in Computer Science, vol. 1141), W. Ebeling, I. Rechenberg, H.-P. Schwefel, and H.-M. Voigt, Eds. Berlin, Germany: Springer-Verlag, 1996, pp. 62–71.

[31] S. J. Louis and G. J. E. Rawlins, "Designer genetic algorithms: Genetic algorithms in structure design," in *Proc. 4th Int. Conf. Genetic Algorithms (ICGA-91)*, R. K. Belew and L. B. Booker, Eds. San Mateo, CA: Morgan Kaufmann, 1991.

[32] M. P. Fourman, "Compaction of symbolic layout using genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms and Their Applica-*

tions, J. J. Grefenstette, Ed. Hillsdale, NJ: Lawrence Erlbaum, 1985, pp. 141–153.

[33] J. Lienig, "A parallel genetic algorithm for performance-driven VLSI routing," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 29–39, Apr. 1997.

[34] G. P. Wagner and L. Altenberg, "Complex adaptations and the evolution of evolvability," *Evolution*, vol. 50, no. 3, pp. 967–976, 1996.

[35] J. R. Koza, D. Andre, F. H Bennett III, and M. A. Keane, "Use of automatically defined functions and architecture-altering operations in automated circuit synthesis with genetic programming," in *Genetic Programming 1996: Proc. 1st Annual Conf. (GP'96)*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds. Cambridge, MA: MIT Press, 1996, pp. 132–140.

[36] J. F. Miller and P. Thomson, "Aspects of digital evolution: Geometry and learning," in *Proc. 2nd Int. Conf. Evolvable Systems (ICES'98)* (Lecture Notes in Computer Science, vol. 1478), M. Sipper, D. Mange, and A. Pérez-Uribe, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 25–35.

[37] A. Thompson, "Evolving electronic robot controllers that exploit hardware resources," in *Advances in Artificial Life: Proc. 3rd Eur. Conf. Artificial Life (ECAL'95)* (Lecture Notes in Artificial Intelligence, vol. 929), F. Morán, A. Moreno, J. J. Merelo, and P. Chacon, Eds. Berlin, Germany: Springer-Verlag, 1995, pp. 640–656.

[38] D. J. Comer, *Digital Logic and State Machine Design*. New York: Holt, Rinehart and Winston, 1984.

[39] A. Thompson, *Hardware Evolution: Automatic Design of Electronic Circuits in Reconfigurable Hardware by Artificial Evolution, Distinguished Dissertation Series*. New York: Springer-Verlag, 1998.

[40] F. P. R. Prosser and D. W. Winkel, *The Art of Digital Design: An Introduction to Top-Down Design*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall 1986.

[41] L. R. Marino, "General theory of metastable operation," *IEEE Trans. Comput.*, vol. C-30, pp. 107–115, Feb. 1981.

[42] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[43] I. Harvey, "Species adaptation genetic algorithms: A basis for a continuing SAGA," in *Toward a Practice of Autonomous Systems: Proc. 1st Eur. Conf. Artificial Life*, F. J. Varela and P. Bourgine, Eds. Cambridge, MA: MIT Press, 1992, pp. 346–354.

[44] I. Harvey, P. Husbands, and D. Cliff, "Genetic convergence in a species of evolved robot control architectures," in *Proc. 5th Int. Conf. Genetic Algorithms*, S. Forrest, Ed. San Mateo, CA: Morgan Kaufmann, 1993, p. 636.

[45] V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology*. Cambridge, MA: MIT Press, 1984.

[46] J.-A. Meyer, P. Husbands, and I. Harvey, "Evolutionary robotics: A survey of applications and problems," in *Proc. 1st Eur. Conf. Evolutionary Robotics (EvoRobot'98)* (Lecture Notes in Computer Science, vol. 1468), P. Husbands and J.-A. Meyer, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 1–21.

[47] O. Holland, "Grey Walter: The pioneer of real artificial life," in *Artificial Life V: Proc. 5th Int. Workshop on the Synthesis and Simulation of Living Systems*, C. Langton, Ed. Cambridge, MA: MIT Press, 1996, pp. 16–23.

[48] T. Higuchi and Y. Hirao, "Evolvable hardware with genetic learning," in *Proc. 2nd Workshop on Synthetic Worlds—Modelizations, Practices, Societal Impacts*, Paris, Jan. 1995.

[49] A. Thompson, "Evolving fault tolerant systems," in *Proc. 1st IEE/IEEE Int. Conf. Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95)*, 1995, pp. 524–529.

[50] A. V. Sebald and D. B. Fogel, "Design of fault tolerant neural networks for pattern classification," in *Proc. 1st Ann. Conf. Evolutionary Programming*, D. B. Fogel and W. Atmar, Eds., EP Society, La Jolla, CA, 1992, pp. 90–99.

[51] W. D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure," in *Artificial Life II*, C. Langton *et al.*, Eds. Reading, MA: Addison-Wesley, 1992, pp. 313–324.

[52] J. Paredis, "Steps toward co-evolutionary classification neural networks," in *Artificial Life IV: Proc. 4th Int. Workshop on the Synthesis and Simulation of Living Systems*, R. Brooks and P. Maes, Eds. Cambridge, MA: MIT Press, 1994, pp. 102–108.

[53] R. Negrini, M. Sami, and R. Stefanelli, *Fault-Tolerance Through Reconfiguration of VLSI and WSI Arrays*. Cambridge, MA: MIT Press, 1989.

[54] D. Mange, M. Goeke, and D. Madon, A. Stauffer, G. Tempesti, and S. Durand, "Embryonics: A new family of coarse-grained field-programmable gate array with self-repair and self-reproducing properties," in *Toward Evolvable Hardware: The Evolutionary Engineering Approach* (Lecture Notes in Computer Science, vol. 1062), E. Sanchez and M. Tomassini, Eds. Berlin, Germany: Springer-Verlag, 1996, pp. 197–220.

[55] A. Thompson, "Silicon evolution," in *Genetic Programming 1996: Proc. 1st Annual Conf. (GP'96)*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds. Cambridge, MA: MIT Press, 1996, pp. 444–452.

[56] L. Huelsbergen, E. Rietman, and R. Slous, "Evolution of a stable multivibrators *in silico*," in *Proc. 2nd Int. Conf. Evolvable Systems (ICES'98)* (Lecture Notes in Computer Science, vol. 1478), M. Sipper, D. Mange, and A. Pérez-Uribe, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 66–77.

[57] Xilinx, Inc., "XC6200 field programmable gate arrays," *Data Sheet*, Version 1.10, Apr. 1997.

[58] I. Harvey and A. Thompson, "Through the labyrinth evolution finds a way: A silicon ridge," in *Proc. 1st Int. Conf. Evolvable Systems (ICES'96)* (Lecture Notes in Computer Science, vol. 1259), T. Higuchi and M. Iwata, Eds. Berlin, Germany: Springer-Verlag, 1997, pp. 406–422.

[59] M. Eigen and P. Schuster, *The Hypercycle—A Principle of Natural Selforganization*. Berlin, Germany: Springer, 1979.

[60] Y. Pomeau and P. Manneville, "Intermittent transition to turbulence in dissipative dynamical systems," *Commun. Math. Phys.*, vol. 74, pp. 189–197, 1980.

[61] D. J. Jefferies and J. H. B. Deane, "Noise, traps, and snags in iterating systems," in *Proc. 1997 Eur. Conf. Circuit Theory and Design (ECCTD'97)*, Budapest, Sept. 1997.

[62] T. Anderson, Ed., *Resilient Computing Systems*. London, U.K.: Collins, 1985.

[63] H. A. Simon, *The Sciences of the Artificial*, 3rd ed. Cambridge, MA: MIT Press, 1996.

[64] T. D. Burton, *Introduction to Dynamic Systems Analysis*. New York: McGraw-Hill, 1994.

[65] R. J. Prance, T. D. Clark, H. Prance, and A. Clippingdale, "Non-contact VLSI imaging using a scanning electric potential microscope," *Meas. Sci. Technol.*, vol. 9, pp. 1229–1235, 1998.

[66] A. Thompson, "Temperature in natural and artificial systems," in *Proc. 4th Eur. Conf. Artificial Life (ECAL'97)*, P. Husbands and I. Harvey, Eds. Cambridge, MA: MIT Press, 1997, pp. 388–397.

[67] S. A. Kauffman, *The Origins of Order*. London, U.K.: Oxford Univ. Press, 1993.

[68] A. Thompson, "On the automatic design of robust electronics through artificial evolution," in *Proc. 2nd Int. Conf. Evolvable Systems (ICES'98)* (Lecture Notes in Computer Science, vol. 1478), M. Sipper, D. Mange, and A. Pérez-Uribe, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 13–24.

[69] P. Layzell, "A new research tool for intrinsic hardware evolution," in *Proc. 2nd Int. Conf. Evolvable Systems (ICES'98)* (Lecture Notes in Computer Science, vol. 1478), M. Sipper, D. Mange, and A. Pérez-Uribe, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 47–56.

[70] P. Layzell, "Reducing hardware evolution's dependency on FPGA's," in *Proc. 7th Int. Conf. Microelectronics for Neural, Fuzzy and Bio-Inspired Systems (MicroNeuro'99)*. Los Alamitos, CA: IEEE Comput. Soc. Press, 1999, pp. 171–178.

[71] D. H. Horrocks and Y. M. A. Khalifa, "Genetically derived filter circuits using preferred value components," in *Proc. IEE Colloq. on Analogue Signal Processing*, Oxford, UK, Oct. 1994, pp. 4/1–4/5.

[72] J. R. Koza, F. H Bennett III, J. Lohn F. Dunlap, D. Andre, and M. A. Keane, "Automated synthesis of computational circuits using genetic programming," in *Proc. IEEE Conf. Evolutionary Computation*. Piscataway, NJ: IEEE Press, 1997, pp. 447–452.

[73] W. Kruiskamp and D. Leenaerts, "DARWIN: CMOS opamp synthesis by means of a genetic algorithm," in *Proc. 32nd Design Automation Conf.*, Assoc. Computing Machinery, 1995, pp. 433–438.

Adrian Thompson received the master's degree in microelectronic systems engineering from the University of Manchester Institute of Science and Technology (UMIST) and the D.Phil. degree from the University of Sussex.

He is currently a Researcher and Lecturer at the University of Sussex in the School of Cognitive and Computing Sciences. Working within the Centre for Computational Neuroscience and Robotics and the Centre for the Study of Evolution, his primary research interest is the application of evolutionary algorithms to design, especially evolutionary electronics.

Paul Layzell received the Master's degree in computer systems engineering at the University of Sussex, UK.

Following subsequent research of distributed algorithms for VLSI at ENSTB, France, he is now a graduate student at the Centre for Computational Neuroscience and Robotics back at Sussex, studying hardware evolution.

Ricardo Salem Zebulum received the B.S. degree in electronic engineering and the M.Sc. degree in computer science in 1992 and 1995, respectively, In 1999, he received the Doctoral degree after performing a joint program involving the Catholic University of Rio and the University of Sussex in England.

His research interests include evolutionary systems applied to electronics, and VLSI implementations of fuzzy logic and artificial neural network algorithms.